A TRIDENT SCHOLAR PROJECT REPORT

NO. 444

Cooperative Control of Unmanned Surface Vessels and Unmanned Underwater Vessels for Asset Protection

by

Midshipman 1/C Gabriel Y.K. Tang, USN

UNITED STATES NAVAL ACADEMY
ANNAPOLIS, MARYLAND

U.S.N.A. --- Trident Scholar project report; no. 444 (2015)

COOPERATIVE CONTROL OF UNMANNED SURFACE VESSELS AND UNMANNED UNDERWATER VESSELS FOR ASSET PROTECTION

by

Midshipman 1/C Gabriel Y.K. Tang
United States Naval Academy
Annapolis, Maryland

_____
(signature)

Certification of Adviser(s) Approval

Professor Bradley E. Bishop
Weapons and Systems Engineering Department

_____
(signature)

_____
(date)

Acceptance for the Trident Scholar Committee

Professor Maria J. Schroeder
Associate Director of Midshipman Research

_____
(signature)

_____
(date)

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
05-18-2015		

4. TITLE AND SUBTITLE
Cooperative Control of Unmanned Surface Vessels and Unmanned Underwater Vessels for Asset Protection

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Tang, Gabriel Ying Kit

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Naval Academy
Annapolis, MD 21402

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Trident Scholar Report no. 444 (2015)

12. DISTRIBUTION / AVAILABILITY STATEMENT

This document has been approved for public release; its distribution is UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This project focused on the development of a control system for a heterogeneous swarm of unmanned surface vessels (USVs) and unmanned underwater vehicles (UUVs) used for asset protection. The control system utilizes a hybrid control scheme, relying on both behavior-based and systems-theoretic concepts. Under this hybrid approach, the swarm is provided with better adaptability, robustness, and overall performance than it would possess under either of these methods alone. Simulations demonstrate the efficacy of the controller for the primary task (asset protection) as well as several secondary tasks.

The first part of this project focused on generating the capability functions, designing the primary and secondary controller and utilizing simulations in different environments to ensure that the controller works as desired. The second part of the project then focused on the hybrid aspect of the swarm, where long baseline technique were used in order to localize the UUV and to mitigate capability degradation in the sub-surface domain. The ability to interdict targets on the surface and the sub-surface was also considered and included as part of the capability control system. The results of this experiment provide a robust, adaptable, and highly mission-capable control system for a cooperative swarm of USVs and UUVs. This in turn will provide the foundation for future systems of a cooperative nature.

15. SUBJECT TERMS
hybrid control, cooperative control, swarm control, redundant manipulator, capability, heterogeneous swarm

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**		119	**19b. TELEPHONE NUMBER** *(include area code)*

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

ABSTRACT

This project focused on the development of a control system for a heterogeneous swarm of unmanned surface vessels (USVs) and unmanned underwater vehicles (UUVs) used for asset protection. The control system utilizes a hybrid control scheme, relying on both behavior-based and systems-theoretic concepts. Under this hybrid approach, the swarm is provided with better adaptability, robustness, and overall performance than it would possess under either of these methods alone. Simulations demonstrate the efficacy of the controller for the primary task (asset protection) as well as several secondary tasks.

Previous swarm control techniques have relied on either an *a priori* formation or a statistical approach wherein the swarm was controlled based on its overall mean and variance. While these approaches allow the user to control the shape or extent of the swarm, they remain limited in that they are not mission-based and take into account the capabilities of the units only insofar as the designer has tuned the system to accommodate them. This project therefore utilizes a *capability function* as the driver for the swarm. This capability function uses real time sensor data in order to measure and allow control of actual mission parameters – for example, the probability of detection of a patrol vessel. Based on the capability desired, the swarm maneuvers itself such that the capability desired is met.

The techniques used to achieve capability-based control accommodate the inclusion of secondary objectives for the swarm. These tasks are defined and carried out in such a way as to allow the primary task to be unaffected, utilizing redundancy that is inherent in a many-unit swarm.

The first part of this project focused on generating the capability functions, designing the primary and secondary controller and utilizing simulations in different environments to ensure

that the controller works as desired. The second part of the project then focused on the hybrid aspect of the swarm, where long baseline technique were used in order to localize the UUV and to mitigate capability degradation in the sub-surface domain. The ability to interdict targets on the surface and the sub-surface was also considered and included as part of the capability control system.

The results of this experiment provide a robust, adaptable, and highly mission-capable control system for a cooperative swarm of USVs and UUVs. This in turn will provide the foundation for future systems of a cooperative nature.

Keywords

Hybrid Control

Cooperative Control

Swarm Control

Redundant Manipulator

Capability

Heterogeneous Swarm

ACKNOWLEDGEMENTS

There are a number of people without whom this Trident Scholar Project would not have been possible – and to those people, I am greatly indebted and would like to express my deepest appreciation and gratitude.

To begin, I am especially grateful to Professor Bradley Bishop, my research advisor. His guidance, support, patience and invaluable knowledge were instrumental in making this project a success thus far. The countless hours of assistance and help are testament to his commitment as an educator. Furthermore, I extend this debt of gratitude to the entire Systems Engineering Faculty, for providing the resources and knowledge so readily.

Special thanks must be given to the Trident Scholar Committee – in particular, Professor Maria Schroeder – for their continued support and assistance in facilitating this Trident Scholar Project.

Lastly, I am eternally grateful to the care and support of my parents, sponsor parents, girlfriend and classmates. Without which, I would not have survived the Naval Academy, let alone complete this project.

“"It is a great profession. There is the fascination of watching a figment of the imagination emerge through the aid of science to a plan on paper. Then it moves to realization in stone or metal or energy. Then it brings jobs and homes to men. Then it elevates the standards of living and adds to the comforts of life. That is the engineer's high privilege.”

- *Herbert C. Hoover*

TABLE OF CONTENTS

Abstract			1
Acknowledgements			3
Table of Contents			4
List of Figures			7
List of Equations			11
Chapter 1	Introduction		13
Chapter 2	Background and Theory		15
Chapter 3	Capability Function		18
	3.1	Overview	18
	3.2	Previous and Current Work	18
	3.3	Capability for Surface Detection through Electro-Optics	19
	3.4	Capability for Surface Detection through Radar	23
	3.5	Capability for Sub-Surface Detection through Active Sonar	27
Chapter 4	Primary Swarm Controller		29
	4.1	Overview	29
	4.2	Controller Matrices	30
	4.3	Overall Capability	31
	4.4	Jacobian Matrix	32
	4.5	Primary Controller Equations	32

Chapter 5	Secondary Swarm Controller		34
	5.1	Overview	34
	5.2	Secondary Objectives	34
	5.3	Artificial Potential Fields Theory	36
	5.4	Final Controller	38
Chapter 6	Surface Simulations		39
	6.1	Overview	39
	6.2	Simulations with Primary Control	40
	6.3	Simulations with Hybrid Control	43
Chapter 7	Sub-Surface Simulations		46
	7.1	Overview	46
	7.2	Simulations with Primary Control	46
	7.3	Simulations with Hybrid Control	48
Chapter 8	Combined Simulations		51
	8.1	Overview	51
	8.2	Simulations with Primary Control	51
	8.3	Simulations with Hybrid Control	53
Chapter 9	Cooperative Localization of UUVs		57
	9.1	Overview	57
	9.2	Previous and Current Works	58
	9.3	Implementation of Cooperative Navigation	60
	9.4	Simulation without Cooperative Navigation	61
	9.5	Simulation with Cooperative Navigation	64

Chapter 10 Interdiction 68

10.1 Overview 68

10.2 Implementation of Interdiction 68

10.3 Interdiction with 3 UUV 1 USV Swarm 69

10.4 Interdiction with 4 UUV 1 USV Swarm 71

Chapter 11 Conclusions 75

11.1 Contributions 75

11.2 Future Works 75

Appendix A Code for Simple Surface Primary Simulation 59

Appendix B Code for Surface Hybrid Control 61

Appendix C Code for Sub-Surface Hybrid Control 65

Appendix D Code for Surface and Sub-Surface Hybrid Control 70

Appendix E Code for Cooperative Navigation 70

Appendix F Code for Cooperative Navigation and Interdiction 70

Bibliography 117

LIST OF FIGURES

Figure 2.1	Table of Comparison of Different Controllers	17
Figure 3.1	Rafael's Toplite III Electro Optic	19
Figure 3.2	Capability Being Delivered as a Function of Distance for the TOPLITE EO	22
Figure 3.3	3D Representation of Capability Delivered for the TOPLITE EO	22
Figure 3.4	Overhead View of Capability Delivered for the TOPLITE EO	22
Figure 3.5	Telephonics RDR-1700A Radar System	23
Figure 3.6	Capability Delivered for the Telephonics RDR-1700A in 2D	25
Figure 3.7	3D Representation of Capability Delivered for the RDR-1700A Radar	26
Figure 3.8	Overhead View of Capability Delivered for the RDR-1700A Radar	26
Figure 3.9	Capability Delivered for Sonar with f = 1.5KHz	28
Figure 4.1	Block Diagram for Primary Controller	29
Figure 4.2:	MATLAB Code to Calculate Overall Capability	31
Figure 5.1	Secondary Objectives for Swarms	35
Figure 5.2	Example of Artificial Potential Field	36
Figure 5.3	Block Diagram for Hybrid Controller	38
Figure 6.1	Two Unit Swarm with One Target	40
Figure 6.2	Change in Capability over Time for Two Unit Swarm	41

Figure 6.3	Four Unit Swarm with Two Targets for Primary Control	41
Figure 6.4	Four Unit Swarm with Two Targets Initial Position	42
Figure 6.5	Four Unit Swarm with Two Targets Final Position	42
Figure 6.6	Change in Capability Over Time for Four Unit Swarm	43
Figure 6.7	Four Unit Swarm with Two Targets for Hybrid Control	44
Figure 6.8	Four Unit Swarm with Two Targets for Hybrid Control at t = 5s	44
Figure 6.9	Capability Delivered Over Time for Hybrid Control	45
Figure 7.1	Front View for Sub-Surface Primary Control of 3 UUVs	47
Figure 7.2	Back View for Sub-Surface Primary Control of 3 UUVs	47
Figure 7.3	Capability Delivered over Time for Sub-Surface Primary Control of 3 UUVs	48
Figure 7.4	Front View for Sub-Surface Hybrid Control of 3 UUVs	49
Figure 7.5	Back View for Sub-Surface Hybrid Control of 3 UUVs	49
Figure 7.6	Capability Delivered over Time for Sub-Surface Hybrid Control of 3 UUV	50
Figure 8.1	Front View for Combined Hybrid Control of 4 USVs and 3 UUVs	52
Figure 8.2	Back View for Combined Hybrid Control of 4 USVs and 3 UUVs	52
Figure 8.3	Capability Delivered over Time for 4 USVs and 3 UUVs	53
Figure 8.4	Position of USVs and UUVs after T = 5s	54
Figure 8.5	Front View for Hybrid Control of 4 USVs and 3 UUVs	54
Figure 8.6	Back View for Hybrid Control of 4 USVs and 3 UUVs	55

Figure 8.7	Capability Delivered over Time for Hybrid Control of 4 USVs and 3 UUVs	55
Figure 9.1	Submersible Utilizing LBL Systems	58
Figure 9.2	CN Algorithm to Calculate Position of UUVs	58
Figure 9.3	Results of CN Algorithm	59
Figure 9.4	Capability Degradation of UUVs	60
Figure 9.5	Initial Simulation Set-up	62
Figure 9.6	Simulation End State (Top View)	62
Figure 9.7	Capability Delivered Over Time without CN	63
Figure 9.8	Initial Simulation	64
Figure 9.9	UUVs Providing Required Capability	65
Figure 9.10	UUV 3 Gets Localized	65
Figure 9.11	UUV 3 Replaces UUV 2’s Position	66
Figure 9.12	UUV 2 Gets Localized	66
Figure 9.13	Capability Delivered Over Time with CN	67
Figure 10.1	Initial Simulation Set-Up	69
Figure 10.2	Final Simulation Set-Up	70
Figure 10.3	Capability over Time for Interdiction with 3 UUVs	70
Figure 10.4	4 UUV 1 USV Simulation with Interdiction (t = 500s)	71
Figure 10.5	4 UUV 1 USV Simulation with Interdiction (t = 700s)	72

Figure 10.6 4 UUV 1 USV Simulation with Interdiction (t = 850s) 72

Figure 10.7 4 UUV 1 USV Simulation with Interdiction (t = 1000s) 73

Figure 10.8 4 UUV 1 USV Simulation with Interdiction (t = 1300s) 73

Figure 10.9 Capability over Time for 4 UUV 1 USV Simulation with Interdiction 74

LIST OF EQUATIONS

Equation 1	Capability Function for Detection via EO	20
Equation 2	Signal-to-Noise Ratio Equation for Radar	24
Equation 3	Capability Function for Detection via Radar	24
Equation 4	Signal-to-Noise Ratio Equation for Sonar	27
Equation 5	Signal-to-Noise Ratio Conversion	27
Equation 6	Capability Function for Detection via Sonar	27
Equation 7	Swarm State Matrix Definition	30
Equation 8	Capability Desired Matrix Definition	30
Equation 9	Current Capability Definition	30
Equation 10	Overall Capability Calculation	31
Equation 11	Jacobian Matrix Definition	32
Equation 12	Change in Capability as a Function of Kinematics of Swarm	32
Equation 13	Definition of Moore Penrose Pseudoinverse	33
Equation 14	Control Equation for Primary Control	33
Equation 15	Null Space Projection for Secondary Control	35
Equation 16	Artificial Potential Field Repulsive Vector for X-Direction (2D)	37
Equation 17	Artificial Potential Field Repulsive Vector for Y-Direction (2D)	37

Equation 18 Artificial Potential Field Repulsive Vector for X-Direction (3D) 37

Equation 19 Artificial Potential Field Repulsive Vector for Y-Direction (3D) 37

Equation 20 Artificial Potential Field Repulsive Vector for Z-Direction (3D) 37

Equation 21 Control Equation for Hybrid Control 38

CHAPTER 1 – INTRODUCTION

Nature has always remained a strong inspiration for the progress of technology. The very idea of swarms – a large number of autonomous decentralized systems working together in cooperation – was inspired by colonies of ants and bees. These large swarms act cooperatively towards achieving a specific objective. Through the use of complex functions and algorithms, mankind is able to artificially create swarms of autonomous systems to simulate those found in nature.

In recent conflicts in the Middle East, robots were increasingly used to take over dangerous and menial tasks such as surveillance, reconnaissance and explosive ordnance disposal. The effectiveness of such autonomous systems is reflected in the fact that militaries all over the world are placing a renewed importance on autonomous systems and robots [1].

A single autonomous robot faces many limitations in terms of its cost effectiveness and spatial area of influence, therefore, making it sub-optimal. However, a group of robots working cooperatively and collectively removes several of the limitations that are imposed on a singular unit. On October 5, 2014, the Office of Naval Research demonstrated a swarm of self-guided unmanned patrol boats that could protect warships and attack potential threats [2]. This was a monumental event as it represents a huge leap for swarming technology. Details of the exact algorithm used and the depth of decision-making of the units are not yet available,

While substantial effort in cooperation of autonomous surface vessels has been carried out, there has been little effort toward heterogeneous swarms of cooperating surface and subsurface autonomous vehicles. The goal of this Trident Scholar Research Project is focused on cooperatively controlling a heterogeneous swarm of robots operating in two different domains – the surface and underwater, simultaneously. Utilizing mission-specific properties of

the individual autonomous robotic units together with a cooperative controller, the swarm can accomplish the overall goals efficiently and cooperatively.

CHAPTER 2 – BACKGROUND AND THEORY

Controlling a swarm of robots is inherently more complicated compared to the control of a singular entity. However, given the correct software and control techniques, the potential of swarms is wide-reaching and revolutionary. This has provided the impetus for the development of robotic swarms, especially in the military domain where the amorphous and distributed nature of a well-controlled cooperative swarm could meet the demands of a constantly adaptive and demanding environment.

The theory behind a swarm's ability to perform in an adaptive environment can be attributed to the fact that a swarm can effectively be treated as a redundant manipulator. In robotics, a redundant manipulator is characterized by having more joint space degrees of freedom than there are dimensions of the end-effector or tool space [3]. For example, the human arm has seven degrees of freedom from the shoulder to the wrist, while the hand operates in a six-dimensional configuration space (three degrees of freedom for position and three for orientation, ignoring the fingers). Thus, a typical human arm has one degree of redundancy, so that there are many possible configurations of the shoulder/elbow/wrist that will place the hand in a specific pose.

Thinking of a group of unmanned surface vessels (USVs) as a single unit, each member contributes two degrees of freedom (considering only position on the surface of the water, for now). If there are fewer task variables than degrees of freedom, the swarm is redundant in the same way that a human arm is redundant. There will be an infinite number of possible configurations for any achievable, non-degenerate (of the primary objective) set of swarm-level objectives. Defining the mathematical representation of the task in an appropriate manner is part of the research in this project.

Using this method, early cooperation controllers focused on getting swarms to a specific mean position with a specific spread (variance) [4]. This approach was only successful in directing the swarm to a particular location and achieving an appropriate distribution of units, but was inadequate in maximizing the potential of a heterogeneous swarm, particularly with respect to the military mission and objective. In order to fully maximize each individual unit's contribution within the swarm, the "capability model" was developed to represent each unit's delivered functionality in a meaningful way [4]. The capability function will be further elaborated in Chapter 3.

There are two generic approaches to swarm control that have been accepted as mainstream – the systems-theoretic approach, and behavior-based approach [5]. The systems-theoretic approach is effectively based on a deterministic control response of the system to closed-form, mathematically-oriented mission specifications (such as maintaining a specific trajectory or holding a particular GPS position). This allows every unit in the swarm to directly achieve a specific outcome, enabling primary mission effectiveness. However, the individual units lack the flexibility and unplanned nature of behavioral approaches. Hence, systems-theoretic methods are unable to react to unanticipated changes in their environment, and deal poorly with rapidly evolving conditions that may require significant re-planning. In order for this form of control to be effective, the parameters of the given environment and mission must be known, or at the very least be highly predictable. Unfortunately, the military domain remains very volatile and unpredictable.

On the other hand, behavior-based approaches are flexible and simple to model; they are based on a set of straightforward stimulus-response behaviors that are triggered by appropriate conditions in a hierarchal scheme. An example of this approach would be a robot that traveled forward unless something was within 6" in front of it, when it would turn in a random direction until the obstacle was no longer seen. This scheme would result in a robot that

could travel successfully through a complex environment with no *a priori* information, but there would be no method by which to determine exactly when or how it might reach a specific point unless the details of the environment were known in advance. Higher order systems can exhibit complex emergent behaviors, made up of rapidly switching simple stimulus-response rules. Behavior-based methods tend to be much easier to develop (initially) than systems-theoretic controllers, but their performance is often unpredictable. This ultimately defeats the demands of the swarm in a military environment since unpredictability in the performance is something that cannot be afforded.

Based on these two approaches, a hybrid controller that incorporates the capability function concept was proposed and developed to account for the flaws of each and to utilize the advantage of each method [6]. This hybrid controller effectively allows the primary fulfillment of a primary task while accommodating other secondary tasks that are encoded in the behavioral structure of the control system. Figure 1 summarizes the comparisons of the different controllers. Due to the clear advantage of the hybrid controller, it was chosen as the controller for this research project. The specifics of the hybrid swarm controller will be elaborated in Chapter 4 and Chapter 5.

Characteristics/ Controllers	***Swarm Coordination***	***Unit Level Decision Making***	***Provable results***
Systems-Theoretic	YES	NO	YES
Behavior-Based	NO* (Cooperation can be achieved, but only through *emergent* behaviors)	YES	NO
Hybrid	YES	YES	YES

Figure 2.1: Table of Comparison of Different Controllers

CHAPTER 3 – CAPABILITY FUNCTIONS

3.1 Overview

A *capability function* is used in the developed approach as the primary driver for the control system of a swarm. Capability is a mathematical representation of the functionality of each unit across the operational space. For example, a camera system delivers capability defined in pixels/meter. The capability changes based on the depth of field of the camera and is defined by the optics of the system. In the swarm controller utilized in this work, the capability function properly defines the overall effectiveness of the swarm based on mission specific properties – effectively defining how important and efficient each individual swarm unit is. Due to the importance of this capability function, real-world data and sensors were used to model the capability function.

For this project, asset protection is the mission. The task to be accomplished by the system is to provide a pre-determined level of sensing and interdiction capability at specified points. Each unit in the swarm will be characterized by its capability to detect/interdict surface threats and underwater threats. The asset protection task will be defined by specifying desired capability at certain points, both on the surface and underwater. These points and capability levels would be determined by an operational analysis of potential threat vectors, and are not discussed herein. Formal *capability* functions will be defined in the sequel, followed by the control architecture that will allow the units to cooperatively provide the specified capability on the target locations.

Five capability models were defined for this project– the capability for detection through electro-optics and radar (above-water threats), the capability for detection via sonar (underwater threats), the capability to interdict a surface vessel, and the capability to interdict a sub-surface

vessel. The interdiction capabilities will be addressed in later chapters. Herein, we discuss only sensing capability.

3.2 Past Work

Evan A. Barnes demonstrated the effectiveness of the capability function in simulations and in actual physical trials [7]. In his project, he utilized simple cameras as sensors and used the Khepera II Mobile Robot, which is essentially a small test robot featuring a Motorola 68331, 25MHz processor, propelled by 2 DC brushed servo motors. Using various test beds, the robotic swarm was able to provide the cooperative sensing capability required in all instances while taking into account a myriad of secondary tasks, such as object avoidance and multiple viewpoint positioning.

Due to the complexity in cooperative control, requiring both unmanned surface vessels (USVs) and unmanned underwater vehicles (UUVs), this Trident Scholar project will not involve physical trials.

3.3 Capability for Surface Detection through Electro-Optics

Electro-optics (EO) makes use of the heat signatures radiating from targets in order to identify them. This eliminates the need for an illuminating source and hence makes EO a very effective surveillance, observation, and targeting system. Most USVs have at least one EO system onboard to aid with detection and targeting.

For the purpose of this project, due to the limitation on the availability of information, the EO capability function was generated based on the system specifications of the TOPLITE III EO that was developed by Rafael Systems for the Israeli Defense Force. The TOPLITE III EO is mounted on the Protector USV and has been deployed by the Republic of Singapore Navy in the Persian Gulf and the Gulf of Aden [8].

Figure 3.1: Rafael's Toplite III Electro Optic

Based on *Johnson's Criteria*, we are able to find the minimum pixels required for the detection of a boat and hence were able to properly define the capability function. Johnson's Criteria describes an image-domain approach to analyzing the ability of observers in order to attain a measurement for performance requirements in terms of detection and recognition [9]. For the sake of this project, the capability function is defined as the probability of detecting a 12 meter by 1 meter target based on a side-view aspect in perfect visibility. Applying this criterion to the EO system, 3 pixels are required in order to recognize the target. The equation for the capability function is defined as follows, where (again), the value is a probability of detection of a 12m x 1m target:

$$c_j = f(D) = \begin{cases} 0, & D > H \\ \sin\left(\frac{\pi}{2} * \frac{\frac{Pixel\_of\_System}{2*D*\tan\left(\frac{FOV}{2}\right)}}{Req\_Pixels}\right), & D < H \end{cases} \quad (1)$$

where the *Pixel_of_System* refers to the number of pixels the sensor has, *FOV* refers to the horizontal field of view of the sensor in radians, *D* refers to the distance of the swarm unit to the point where the capability is desired and *Req_Pixels* refers to the number of pixels required for recognition as laid out in Johnson's Criteria. *H* refers to the distance to the horizon.

Effectively, this capability function is based on the probability of detection as determined by the resolution of the system, which is in turn a function of *D*. However, if the distance to the

object is past the horizon, represented by *H*, it would be impossible for the EO to detect the ship. Hence, this accounts for the sudden drop of capability to zero past a certain point.

Based on this equation, a visual representation of how the capability delivered varies over a certain distance is as shown in Figure 3.2.

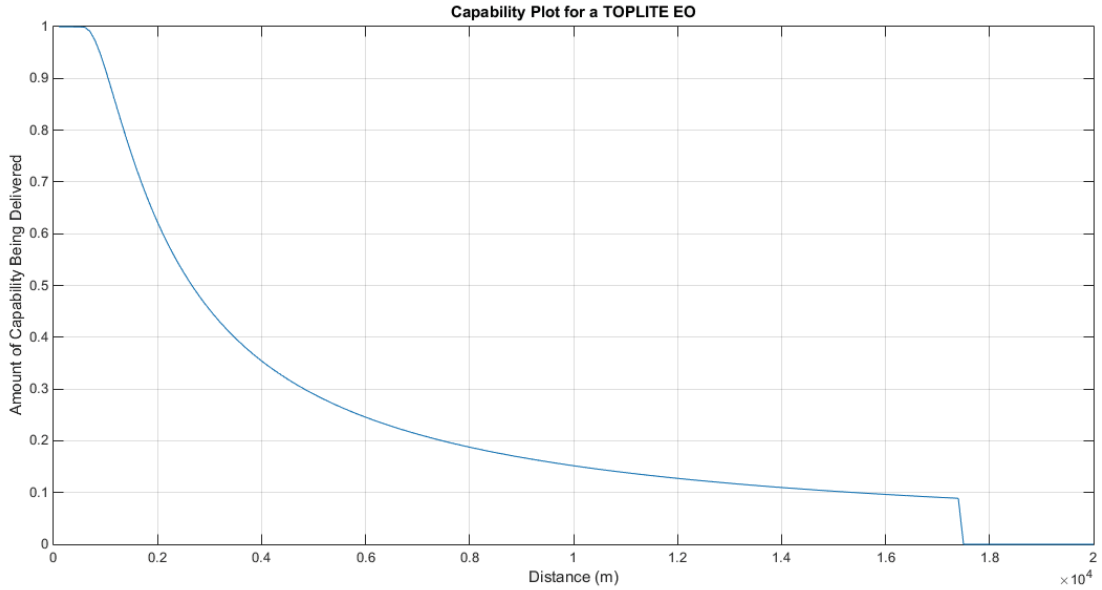

Figure 3.2: Capability Being Delivered as a Function of Distance for the TOPLITE EO (Capability is defined as the probability of detection for a 12m x 1m profile)

Once the capability of an individual swarm unit equipped with an EO wa5s defined, the next step was to provide and analyze this capability from a 3-dimensional perspective. Most USVs have the ability to rotate their EO in order to search the surroundings for target. While this rotation of the EO is not instantaneous, this is hard for the capability function to model since the search pattern of the EO is erratic and does not follow a fixed pattern. However, the time constants associated with a full sweep of the EO are much, much smaller than the time constants associated with surface threats. As such, without loss of generality, the EO capability is assumed to be omnidirectional. Taking that into account, the “capability volcano” of the EO, assuming full radial coverage, is as follows in Figure 3.3 and Figure 3.4.

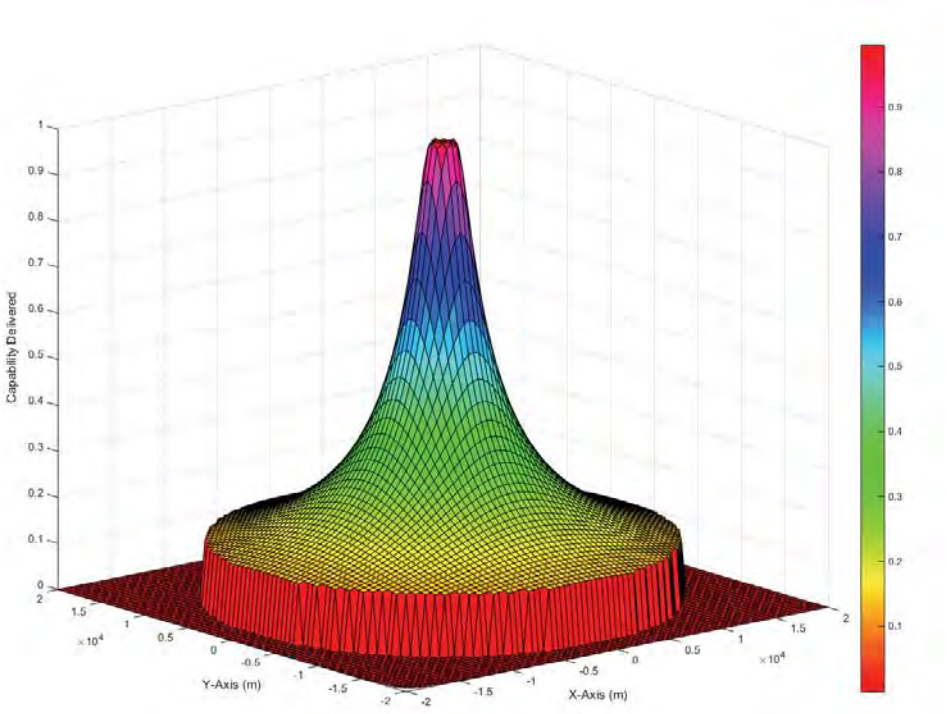

Figure 3.3: 3D Representation of Capability Delivered for the TOPLITE EO

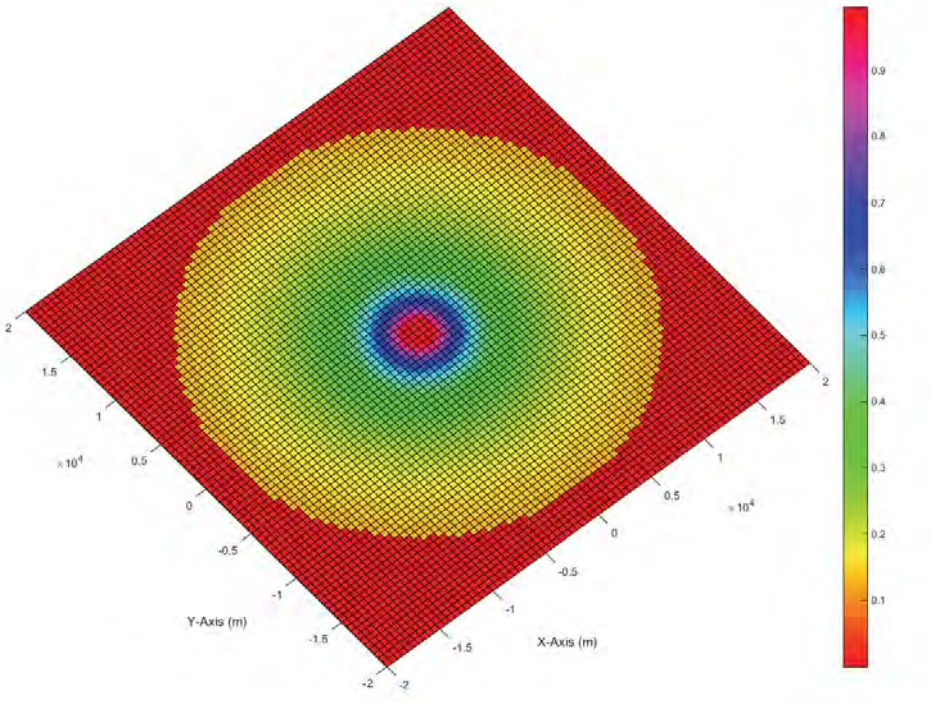

Figure 3.4: Overhead View of Capability Delivered for the TOPLITE EO

3.4 Capability for Surface Detection through Radar

USVs often come equipped with radars in order to further enhance their detection, identification and targeting capability. Radars utilize radio waves in order to determine the range, altitude, direction or speed of objects. By transmitting and “bouncing radio waves” off our target object, we can obtain information about targets in range of the system. Due to the ability to detect targets despite poor visibility, radars have proven to be a valuable assets onboard USVs.

The radar unit selected for the capability model herein is a commercial off-the-shelf (COTS) Telephonics RDR-1700A [10]. The RDR-1700Aa is a lightweight, X-band, and 360-degree search radar. While it is mainly designed for fixed or rotary-wing aircraft, it can also be utilized by marine vessels in order to conduct patrol, surveillance, rescue and precision terrain mapping. The RDR-1700A is as shown in Figure 3.5.

Figure 3.5: Telephonics RDR-1700A Radar System

Due to the myriad of factors that influences the radar's probability of detection, there is a certain complexity in attaining the capability function. However, by utilizing the signal-to-noise ratio in the radar range equation [11] along with some applied statistics [12], the radar detection capability function is defined as follows:

$$SNR = \frac{P_t G_t G_r \sigma \lambda^2}{(4\pi^3) R^4 L} \tag{2}$$

$$c = f(SNR, pfa) = \int_{\sqrt{-2\ln(pfa)}}^{\infty} x\, I_0(x\sqrt{2SNR})\, e^{\left(\frac{-(x^2+2(SNR))}{2}\right)}\, dx \tag{3}$$

where P_t is the peak transmit power in watts, G_t is the gain of the transmitter in decibels, G_r is the gain of the receiver in decibels, and λ is the wavelength in meters of the transmitted radio wave. These are all variables that are defined by the radar itself. The variable *R* represents the distance from the radar to the target while σ is the radar cross section of the target – assumed to be 12 square meters for the purpose of this project.

Through the use of Marcum's Q-function, Equation 3 gives us the probability of detection as a function of the signal-to-noise ratio (*SNR*) as well as the probability of false alarm (*pfa*) [12], which is assumed to be 0.01. I_0 is a modified Bessel function that accounts for the signal processing. Based on these equations, we obtain the visual representation of the capability over distance in Figure 3.6.

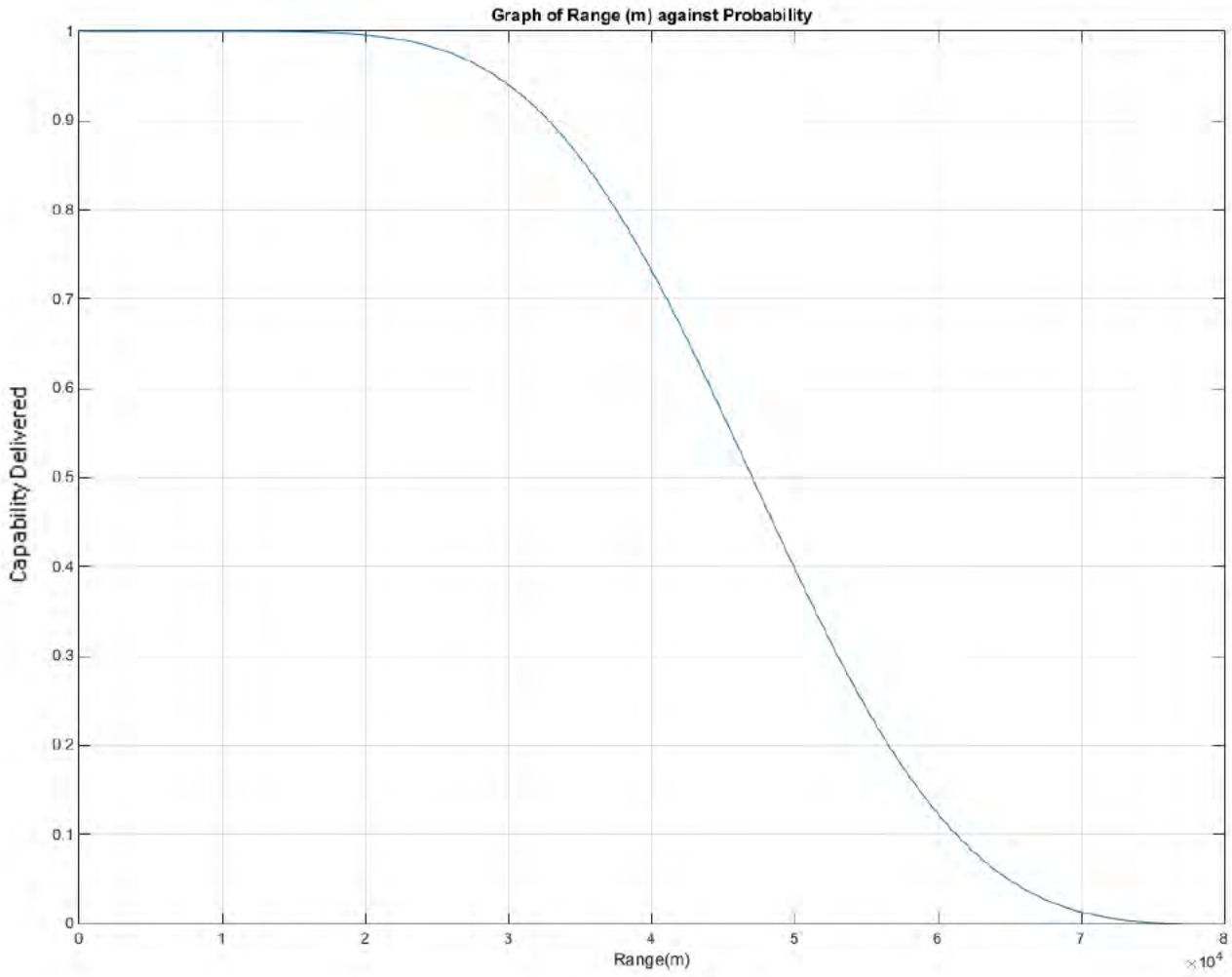

Figure 3.6: Capability Delivered for the Telephonics RDR-1700A in 2D

Similar to the EO, there is a certain field-of-view associated with the radar. However, it too has the ability to rotate and scan the surroundings for targets. As such, it is again assumed that the capability is omni-directional for the Telephonics RDR-1700A radar. With that consideration, the capability volcano for the radar is as shown in Figure 3.7 and 3.8.

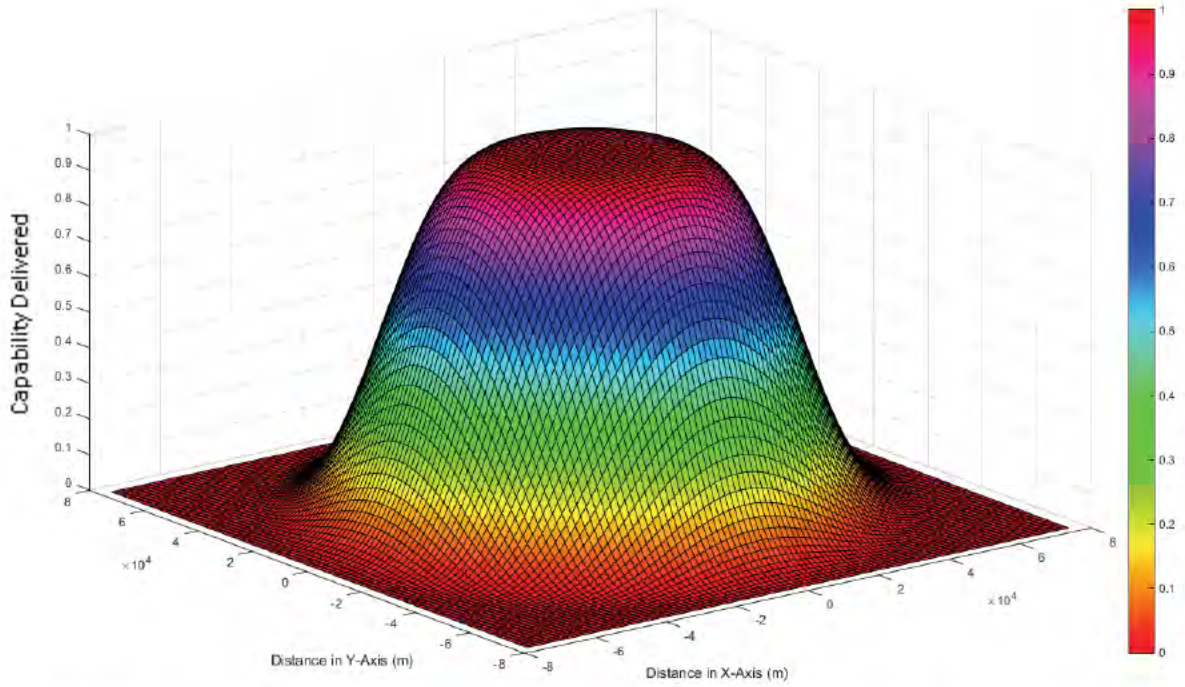

Figure 3.7: 3D Representation of Capability Delivered for the RDR-1700A Radar

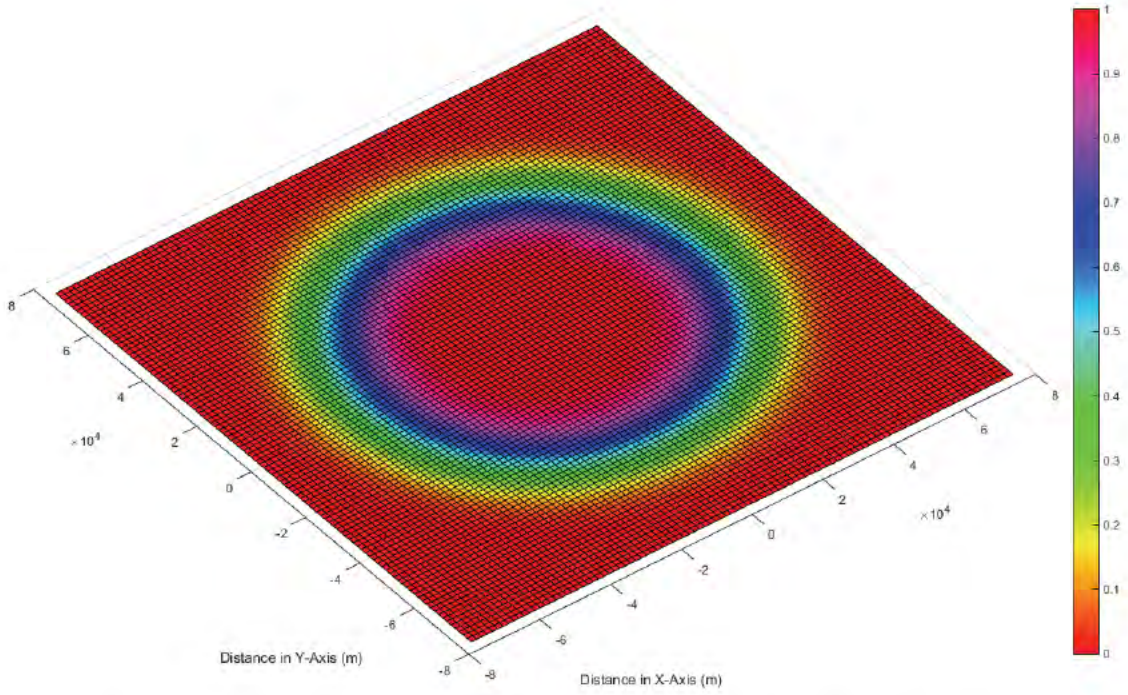

Figure 3.8: Overhead View of Capability Delivered for the RDR-1700A Radar

3.5 Capability for Sub-Surface Detection through Active Sonar

Sonar is a technique where sound waves are utilized in order to navigate, communicate, or detect objects under the surface of the water. An active sonar works by propagating sounds waves via a source that is usually onboard the USV. As acoustic impulses hit the target, they get deflected back to the receiver for analysis. Despite its simplistic nature, sonar remains one of the best and most reliable techniques for sub-surface detection.

Most USVs are equipped with side scan sonar in order to fulfill the task requirements of underwater mapping and detection of mines. However, for the purpose of this research project, each USV must be equipped such that it can detect submarines (and other USVs), as they present the most dangerous threat to the protected asset from the sub-surface domain. For this purpose, an active sonar is best suited for the requirements of the mission.

Similar to the radar, the active sonar's ability to detect a target is a function of both the signal-to-noise ratio (*SNR*) and the probability of false alarm (*pfa*). The equations governing the capability function is shown in Equation 4, 5 and 6 [13], based on a target modeled as a sphere of 10m radius.

$$SNR\ (db) = SL - 2TL + TS \tag{4}$$

$$SNR = 10^{\frac{SNR(db)}{10}} \tag{5}$$

$$c = f(SNR, pfa) = 0.5\left(\frac{2}{\sqrt{\pi}}\int_{erfc^{-1}(2pfa)-\sqrt{SNR}}^{\infty} e^{-x^2} dx\right) \tag{6}$$

SL refers to the source level of the sonar in decibels (dB). This is the strength of the sonar transmission and is a fixed value dependent on the type, efficiency and power of the sonar used. *TL* refers to the reduction in signal intensity, given in decibels (dB). *TL* is therefore a function of range and frequency and is based on the assumption of cylindrical spreading [14].

TS is the target strength which represents the echo returned by an underwater target in decibels. Based on the equations above, the capability plot of a 1.5 kHz sonar is obtained in Figure 3.9.

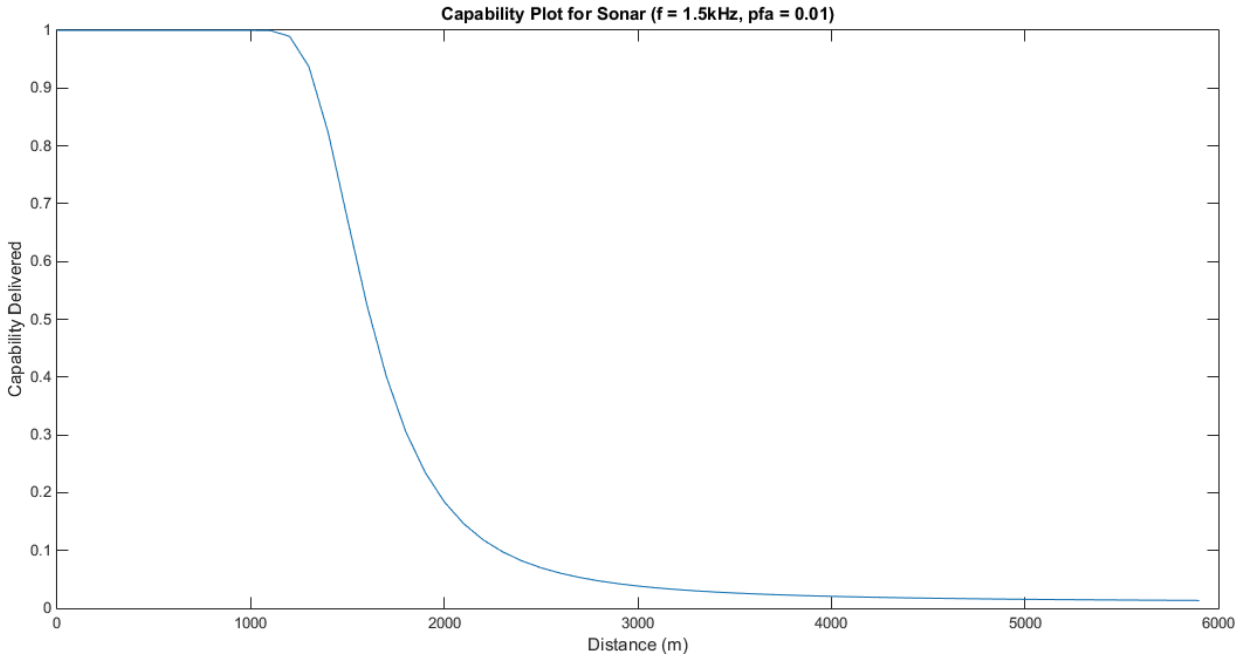

Figure 3.9: Capability Delivered for Sonar with f = 1.5KHz, assuming a target sphere of 10m radius.

Given that sonars operate in the sub-surface domain, the 3D representation of the capability cannot be rendered, but it is again assumed that the system is omnidirectional. In the case of a sonar, this means sensing occurs across the full 3D space (underwater) with capability defined merely by distance from the unit.

CHAPTER 4 – PRIMARY SWARM CONTROLLER

4.1 Overview

The primary swarm controller drives the entire system based on the quantified capability function that is derived in Chapter 3. The primary swarm controller utilizes a systems-theoretic approach where the capability function provides the framework for the system. Systems-theoretic methods are commonly based on differential equations and makes use of compensators in order to give results that match desired specification.

The primary swarm controller that was designed for this project utilizes the Jacobian matrix (which will be further elaborated upon) and is as shown in Figure 4.1.

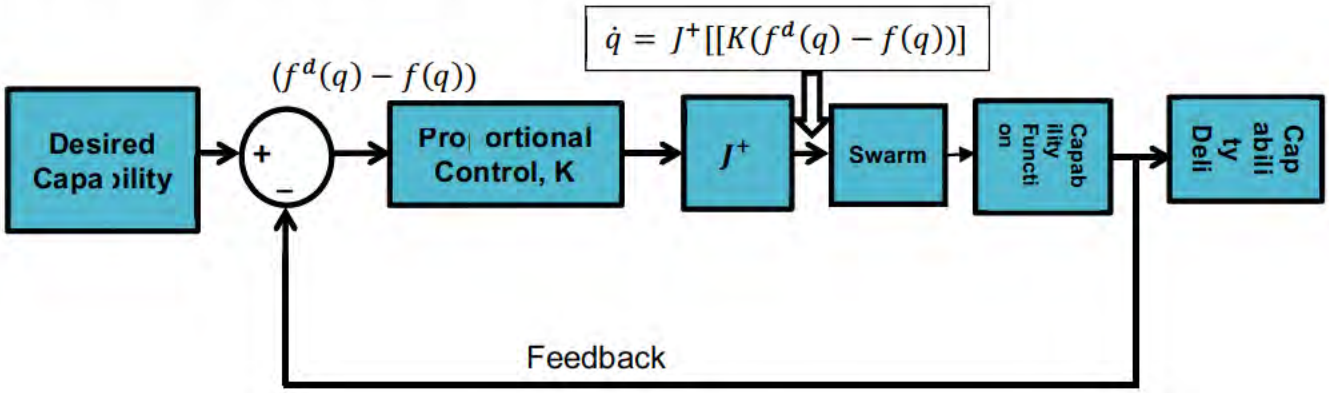

Figure 4.1: Block Diagram for Primary Controller

The desired capability block is the only input into the controller. The desired capability is a vector of detection probabilities desired at a set of points, and is denoted as $f^{d}(q)$, where q is the swarm state (unit positions). The vector of desired detection probabilities is aligned with a set of target positions that are encoded into the controller. Based on the target and the amount

of capability desired by the user, the primary controller will utilize the proportional control and the Moore-Penrose pseudoinverse of the Jacobian to deliver the capability desired.

4.2 Controller Matrices

The controller matrices are the core of the system, and are designed to allow easy mapping between the desired capability and allowable motions of the units [3]. The initial matrices for the system is as defined below, where we are assuming that the units are position-controllable on a scale sufficient to not require any more in-depth kinematics:

$$\textit{Swarm State} = q = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ \vdots \\ x_n \\ y_n \\ z_n \end{bmatrix} \tag{7}$$

$$\textit{Capability Desired} = \begin{bmatrix} c\_d_1 & x\_d_1 & y\_d_1 & z\_d_1 \\ \vdots & \vdots & \vdots & \vdots \\ c\_d_m & x\_d_m & y\_d_m & z\_d_m \end{bmatrix} \tag{8}$$

$$\textit{Current Capability} = f(q) = \begin{bmatrix} c_1 \\ \vdots \\ c_m \end{bmatrix} \tag{9}$$

Equation 7 defines the swarm state which provides the *x*, *y* and *z* location of every unit within the swarm where *n* is the number of units in the swarm. Equation 8 lists the capability desired at *m* specified target locations (columns 2 – 4 define the target points) while Equation 9 provides the current capability on the *m* points. The capability is a function of the swarm state since the capability provided to the target depends on the distance the swarm unit is away from the desired target location and how many units possess appropriate capability.

4.3 Overall Capability

Since the capability function is calculated for each unit within the swarm, it is not sufficient to provide the individual capability at a given point. The capability function is a direct reflection of the probability of detection; hence, the overall capability at a given point can be calculated using the law of total probability [15]. The overall capability is defined as the probability that the target will be detected by at least one unit.

$$\textit{Overall Capability} = P(D) = p_1 + (1 - p_1)p_2 + (1 - (1 - p_1)p_2)p_3 + \cdots \quad (10)$$

where p_i is the probability of detection of a target by unit *i.* Hence, the overall capability can be calculated based on equation above.

```
capability(i) = 0 ;
cadd = 1;
for i = 1:1:length(cap_desired)
    for n = 1:1:((length(swarmstate))/2)
        capability(i) = capability(i) + c(n)*cadd;
        cadd = (1 - capability(i));
    end
end
```

Figure 4.2: MATLAB Code to Calculate Overall Capability

Figure 4.2 shows the above equation translated into MATLAB code. The c*apability* parameter refers to the individual capability a unit provides at the target capability position defined by *i* while *n* refers to the amount of units in the swarm.

4.4 Jacobian Matrix

The controller now has the ability to track every unit's position through the S*warm State* matrix as well as the current capability provided in the C*urrent Capability* matrix. However, it lacks the connection between the task (desired changes in capability) and the current swarm state. The Jacobian matrix thus provides a relationship between the capability and the kinematics of the swarm. The Jacobian matrix is represented in Equation 11 and is effectively a first order derivative of the capability function with respect to the individual state variables. This represents how a change in the *X, Y* or *Z* position of each unit of the swarm unit would affect the capability at the target point.

$$J = \begin{bmatrix} \frac{\delta C_1}{\delta X_1} & \frac{\delta C_1}{\delta Y_1} & \frac{\delta C_1}{\delta Z_1} & \cdots & \frac{\delta C_1}{\delta X_n} & \frac{\delta C_1}{\delta Y_n} & \frac{\delta C_1}{\delta Z_n} \\ & \vdots & & \ddots & & \vdots & \\ \frac{\delta C_m}{\delta X_1} & \frac{\delta C_m}{\delta Y_1} & \frac{\delta C_m}{\delta Z_1} & \cdots & \frac{\delta C_m}{\delta X_n} & \frac{\delta C_m}{\delta Y_n} & \frac{\delta C_m}{\delta Z_n} \end{bmatrix} \tag{11}$$

In Equation 11, *n* refers to the number of units within the swarm while *m* refers to the number of target locations where capability is defined. Due to the complexity of the partial derivatives of the capability function, the Jacobian function in MATLAB is utilized to calculate the Jacobian matrix for the swarm, with the results excluded for brevity.

4.5 Primary Controller Equations

Using the definition of the Jacobian matrix, a change in capability can be calculated based on the motion and kinematics of the units within the swarm according to Equation 12 as follows:

$$\begin{bmatrix} \dot{c_1} \\ \vdots \\ \dot{c_m} \end{bmatrix} = J * \begin{bmatrix} \dot{x_1} \\ \dot{y_1} \\ \dot{z_1} \\ \vdots \\ \dot{x_n} \\ \dot{y_n} \\ \dot{z_n} \end{bmatrix} \tag{12}$$

The controller ultimately needs to move the units within the swarm, and the above equation is not sufficient to provide a control equation for the primary controller. Therefore, the inverse of the Jacobian must be utilized. However, it is key to note that the Jacobian matrix is not necessarily a square matrix. As such, the Moore Penrose pseudo-inverse of the Jacobian matrix has to be taken as follows:

$$J^{+} = J^{T}(JJ^{T})^{-1} \tag{13}$$

Utilizing both Equation 12 and 13 and a proportional gain controller, one can now derive the control for the motions of the units within the swarm as follows in Equation 14.

$$\dot{q} = \begin{bmatrix} \dot{x_1} \\ \dot{y_1} \\ \dot{z_1} \\ \vdots \\ \dot{x_n} \\ \dot{y_n} \\ \dot{z_n} \end{bmatrix} = J^{+} * \begin{bmatrix} \dot{c_1} \\ \vdots \\ \dot{c_m} \end{bmatrix} = J^{+}[K(\begin{bmatrix} c_{1\_d} \\ \vdots \\ c_{m\_}d \end{bmatrix} - \begin{bmatrix} c_1 \\ \vdots \\ c_m \end{bmatrix})] \tag{14}$$

Equation 14 effectively translates into the block diagram in Figure 4.1. By solely utilizing the primary controller, the swarm is able to provide capability on target, as will be seen in the simulations in Chapter 6. It is important to note that the controller now solely uses a system-theoretic approach where the individual swarm units lack any intelligence in the form of decision making abilities.

CHAPTER 5 – SECONDARY SWARM CONTROLLER

5.1 Overview

With the primary controller, the swarm utilizes the systems-theoretic approach in order to attain the capability desired. However, the units within the swarm lack any intelligence in the sense that they do not have any sort of decision making ability at all. In order to introduce a certain level of intelligence into the swarm units, we incorporate behavioral methods into the controller, effectively making it a hybrid controller [16].

Hybridization of the controller again relies on the analogy to a redundant robotic manipulator. Recall, a redundant robotic manipulator is characterized by having more joint space degrees of freedom than there are task space degrees of freedom [3]. For the swarm under consideration, each unit will provide two (in the case of a USV) or three (in the case of a UUV) degrees of freedom, while each capability target point will define one task space degree of freedom. The primary controller uses all of the units to achieve the desired capability, but if there are more unit degrees of freedom than there are capability target points, the system has an infinite number of configurations that will accomplish the desired capability. The extra degrees of freedom can be controlled using a secondary method to achieve additional goals under the behavior-based approach. Ultimately, this allows the controller increased flexibility in fulfilling the capability desired.

5.2 Secondary Objectives

Since the control system is able to take into account behavioral control through the addition of a secondary (redundancy-based) control approach, it is important to consider the secondary objectives for the swarm to achieve.

USVs Tasking	UUVs Tasking
Avoid obstacles	Avoid Obstacles
Avoid Other USVs	Avoid other UUVs
Avoid UUVs on surface	Stay submerged if possible

Figure 5.1: Secondary Objectives for Swarms

The above table represents the minimum consideration for the swarm to operate efficiently. Ideally, the swarm would be able to incorporate military tactics and other elements to obtain an advantage over the adversary through properly defined secondary objectives. However, for the purpose of this project, only the above objectives have been considered at this point. Additional secondary objectives, such as swarm variance, USV-UUV attraction, etc., will be considered in the latter part of the project.

The secondary controller utilizes a gradient projection technique in the form of the null space projection in order to achieve the secondary objectives. The null space control effectively projects the secondary task gradient onto the *null space* of the primary task Jacobian. That is, the secondary task is accomplished using resources coordinated such that they do not influence the primary task. Here, units move in a coordinated manner to avoid obstacles and other units while still maintaining the desired capability. The mathematics of the system guarantee that the primary task is accomplished if possible, and that the secondary tasks are achieved as well as possible without degrading the primary task. The basic control equation for null space projection is shown in Equation 15.

$$\dot{q}_{secondary} = (I - J^{+}J)\,\mathrm{v} \tag{15}$$

The null space projection is defined by the $(I - J^{+}J)$ term and it projects the equation onto the primary task Jacobian. The driving vector for the secondary control, *v,* must be

generated from the behavioral control within the units in the swarm. Artificial potential field theory will be utilized to generate this vector and will be elaborated in Chapter 5.3.

5.3 Artificial Potential Field Theory

Artificial potential field (APF) techniques draw from potential field theory concept in physics, and are used in mobile robotics extensively. Controls are developed to force a robot to behave as a charged particle moving in a potential field, where an attractive force is placed at a target location and repulsive forces are placed around obstacles [17]. This effectively creates a vector field under which an end-effector or a swarm unit would move, skirting around obstacles and moving toward a target location. An example of how artificial potential field works is as seen in Figure 5.3

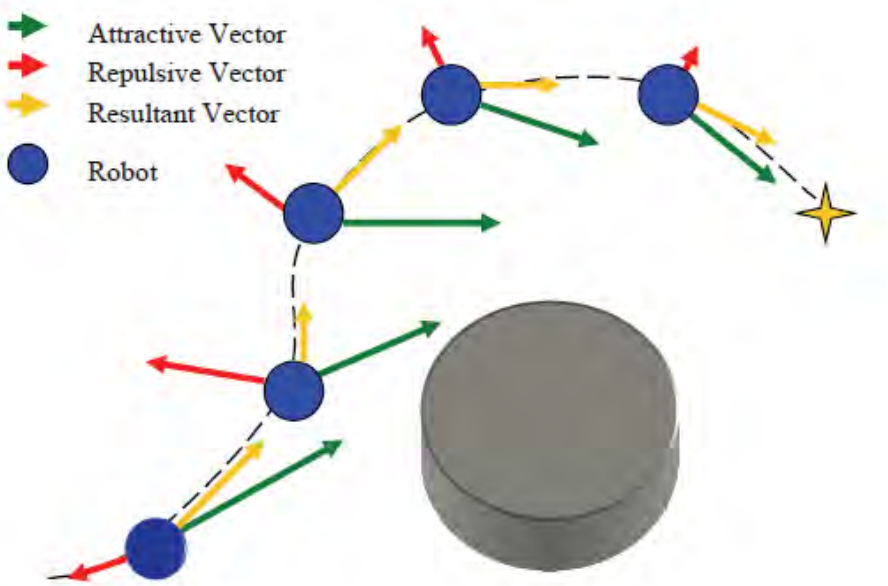

Figure 5.2: Example of Artificial Potential Field [7]

For the purpose of this project, the following repulsive vector was used to achieve inter-unit collision avoidance (no attractive vector is used at this point).

$$V_{repel_x} = V_{repel_x} + K\left(\frac{1}{distance} - \frac{1}{safezone}\right)(\cos\alpha) \quad (16)$$

$$V_{repel_y} = V_{repel_y} + K\left(\frac{1}{distance} - \frac{1}{safezone}\right)(\sin\alpha) \quad (17)$$

$$\alpha = \tan^{-1}\left(\frac{\Delta y}{\Delta x}\right)$$

In the equations above, the repulsive vectors generated are for a 2D system. K is a gain used to control the repulsive vector, while *distance* refers to the distance between the target and the unit and *safezone* refers to the distance to the obstacle at which the secondary repulsive vector will start to be applied. Depending on how Δy and Δx is defined, V_{repel_x} and V_{repel_y} can be utilized for avoiding obstacles, avoiding other units within the swarm or even to avoid the entire surface of the water for underwater vessels. *K* is the control gain for the system and can be manipulated to control how much influence the secondary control have on the overall system. The parameter d*istance* refers to the distance the swarm is away from the obstacle or unit to be avoided while *safezone* is a defined parameter used to control the distance the swarm unit is to avoid the obstacle or unit by. Similarly, Equations 18, 19 and 20 are the repulsive vector generated for a 3D system

$$V_{obs_x} = V_{obs_x} + K\left(\frac{1}{distance} - \frac{1}{safezone}\right)(\cos\alpha + \cos\theta) \quad (18)$$

$$V_{obs_y} = V_{obs_y} + K\left(\frac{1}{distance} - \frac{1}{safezone}\right)(\sin\alpha + \cos\beta) \quad (19)$$

$$V_{obs_z} = V_{obs_z} + K\left(\frac{1}{distance} - \frac{1}{safezone}\right)(\sin\beta + \sin\theta) \quad (20)$$

$$\alpha = \tan^{-1}\left(\frac{\Delta y}{\Delta x}\right)$$

$$\beta = \tan^{-1}\left(\frac{\Delta z}{\Delta y}\right)$$

$$\theta = \tan^{-1}\left(\frac{\Delta z}{\Delta x}\right)$$

5.4 Final Controller

Combining the primary and secondary control, we are able to generate the final control system for the hybrid control as seen in Figure 5.3.

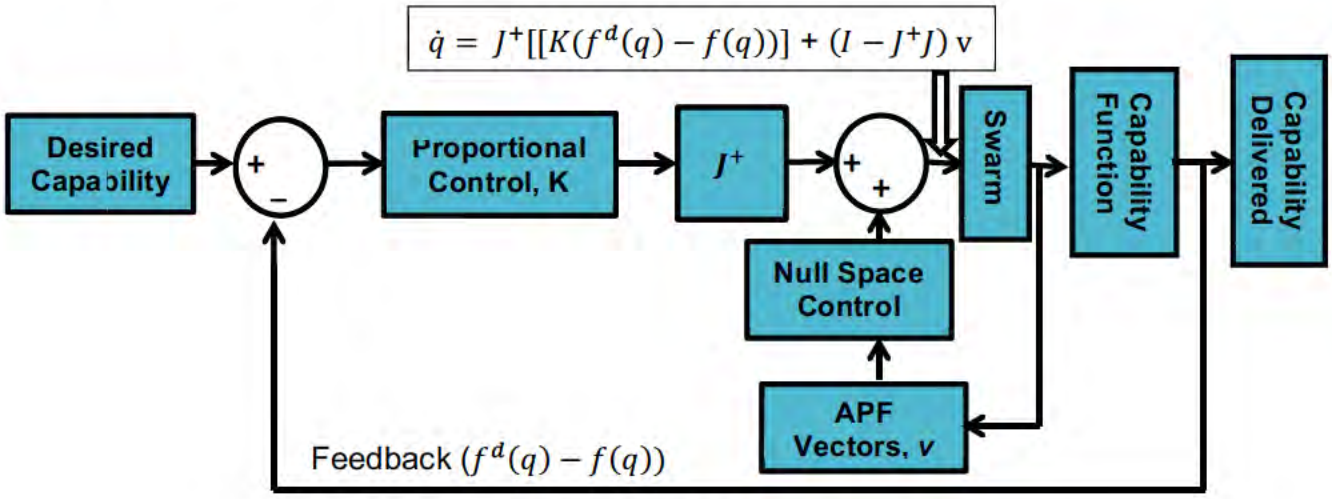

Figure 5.3: Block Diagram for Hybrid Controller

With both systems-theoretic and behavioral control incorporated into the system, the final equation for the control systems is as shown in Equation 21.

$$\dot{q} = J^{+}[K_{pri}(f^{d}(q) - f(q))] + K_{sec}(I - J^{+}J)\,v \tag{21}$$

The variable *v* in the above equation is the vector sum of all the repulsive vectors generated from the secondary objectives listed in Figure 5.1. Furthermore, it is important to note that the control gains of the system, K_{pri} and K_{sec}, can be altered to decide how much influence the primary or secondary control have on the entire system.

CHAPTER 6 – SIMULATION FOR SURFACE VESSELS

6.1 Overview for Simulation

The simulation for the swarm control was carried out in MATLAB, a multi-paradigm numerical computing environment that was developed by MathWorks. The goal of the simulation phase was to design and modify the controller such that it suits the practicality of whatever aspect the mission needs in real-time. However, simulations are limited both in terms of software and hardware. As such, several assumptions were made.

Firstly, the simplifying assumption is made that the swarm unit is holonomic in nature. Holonomic vehicles are defined by the fact that all of their degree of freedom are controllable [18]. Under this definition, a real-world USV or UUV faces non-holonomic constraints, as they are not able to move horizontally without moving in the forward or reverse direction. While most vehicles are non-holonomic in nature, the above assumption can still hold valid if the state to be controlled is defined as the position of a point on the vehicle that is off the common wheel/thruster axis [19]. Additionally, the scales involved in the simulation are significantly larger than the turning radii of the units (tens of nautical miles versus tens of meters), making the short-distance maneuvering control of little significance in the performance of the overall system.

Secondly, we assume that there are sensors aboard the USVs or UUVs that are able to detect the obstacles or each other from a certain distance as defined by the *safezone* variable in Equations 16 to 20. Based on these equations, the repulsive vector generated from the artificial potential field would not take into effect until the unit enters the *safezone.* This assumption is reasonable given that the safe zone can be defined such that it falls under the detection range of the sensors that are utilized.

Lastly, the USV was assumed to be only equipped with EO while the UUVs were assumed to be only equipped with active sonars. The use of additional capabilities would simply require using the law of total probabilities as stated in Equation 10.

For the purpose of the surface simulations, the control gain for the primary control, K_{pri}, was set at 0.09 while the control gain for the secondary control, K_{sec}, was set at 300. The *safezone* for obstacle and unit avoidance was set to 100 meters while the gain for the artificial potential field vector was set to 300. The gains were obtained experimentally and based on observation on the swarm's performance. Furthermore, these gains were chosen in order to ensure that the movement of the swarm units be as realistic as possible.

6.2 Simulation with Primary Control

The simulations were first run at the most simplistic level to ensure that the primary control would work as desired. Figure 6.1 represents a swarm of two units that only utilizes the primary control while Figure 6.2 represents the capability change over time for the swarm. All of the simulations parameters are defined as in the diagrams.

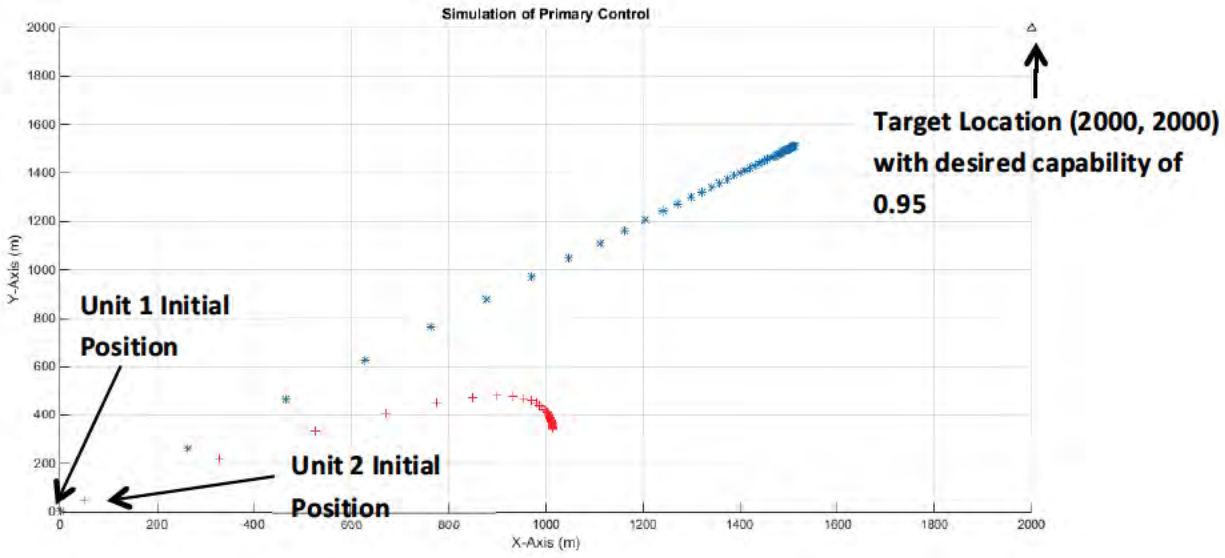

Figure 6.1: Two Unit Swarm with One Target

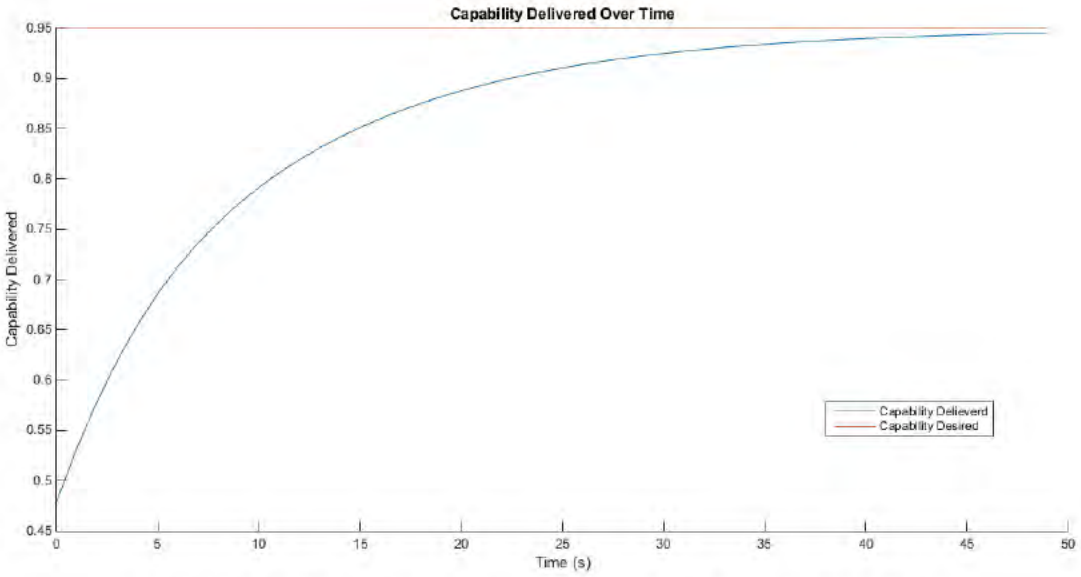

Figure 6.2: Change in Capability over Time for Two Unit Swarm

In Figure 6.1 and 6.2, the effectiveness of the primary control was demonstrated for a simple two unit swarm with one desired capability location. In the following simulation (in Figure 6.3 to 6.6) obstacles were introduced for a four unit swarm with two desired capability locations.

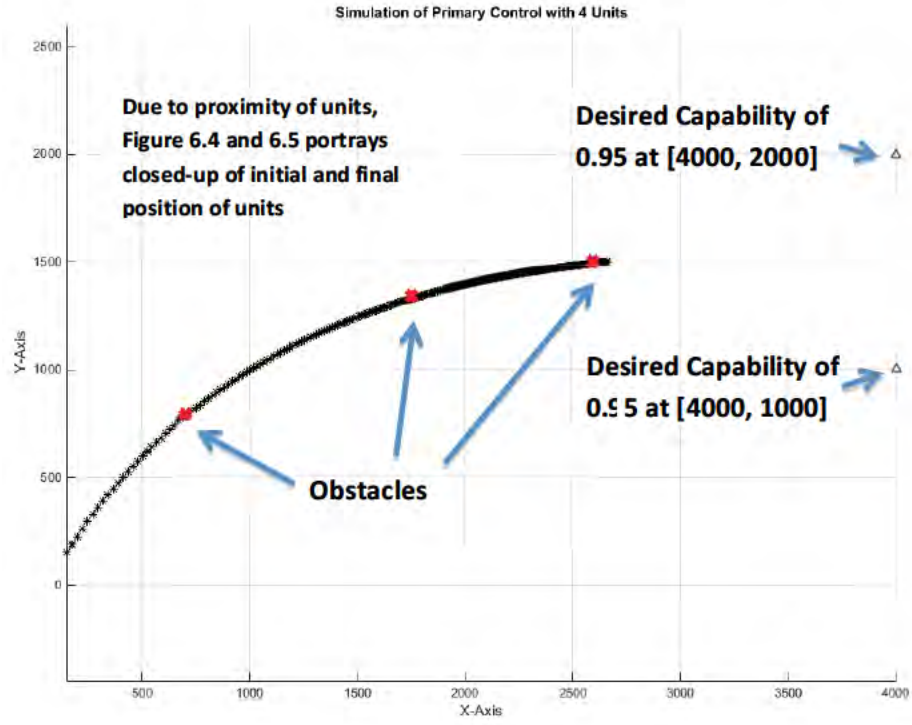

Figure 6.3: Four Unit Swarm with Two Targets for Primary Control

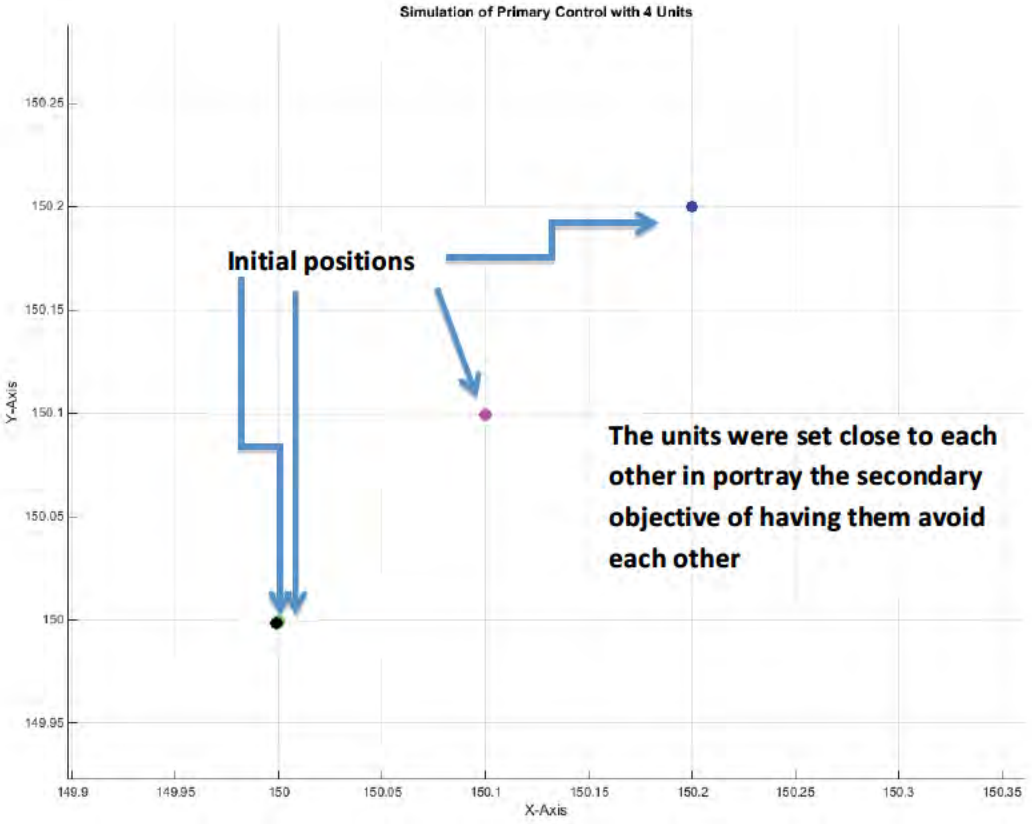

Figure 6.4: Four Unit Swarm with Two Targets Initial Position (t = 0s)

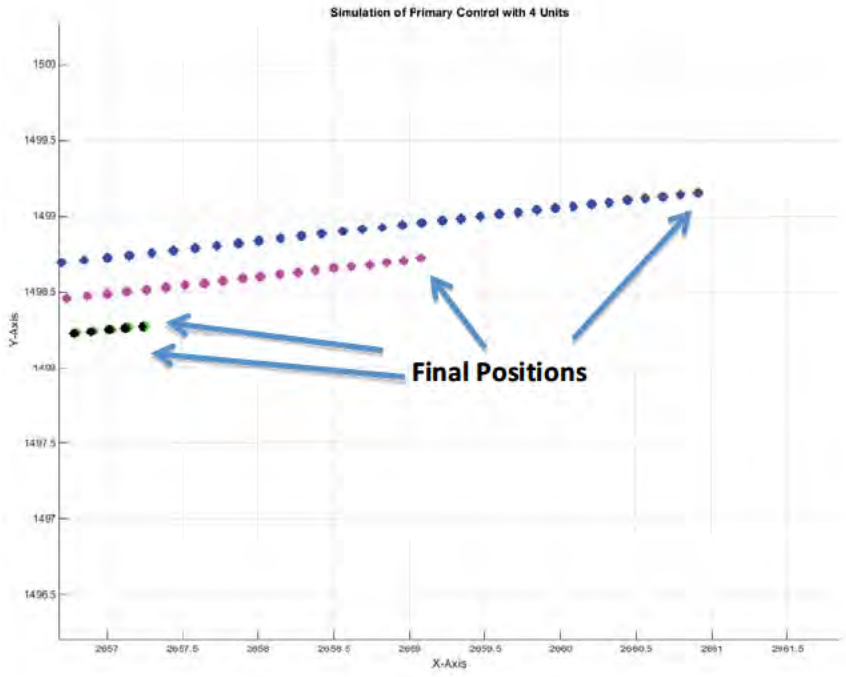

Figure 6.5: Four Unit Swarm with Two Targets Final Position (t = 600s)

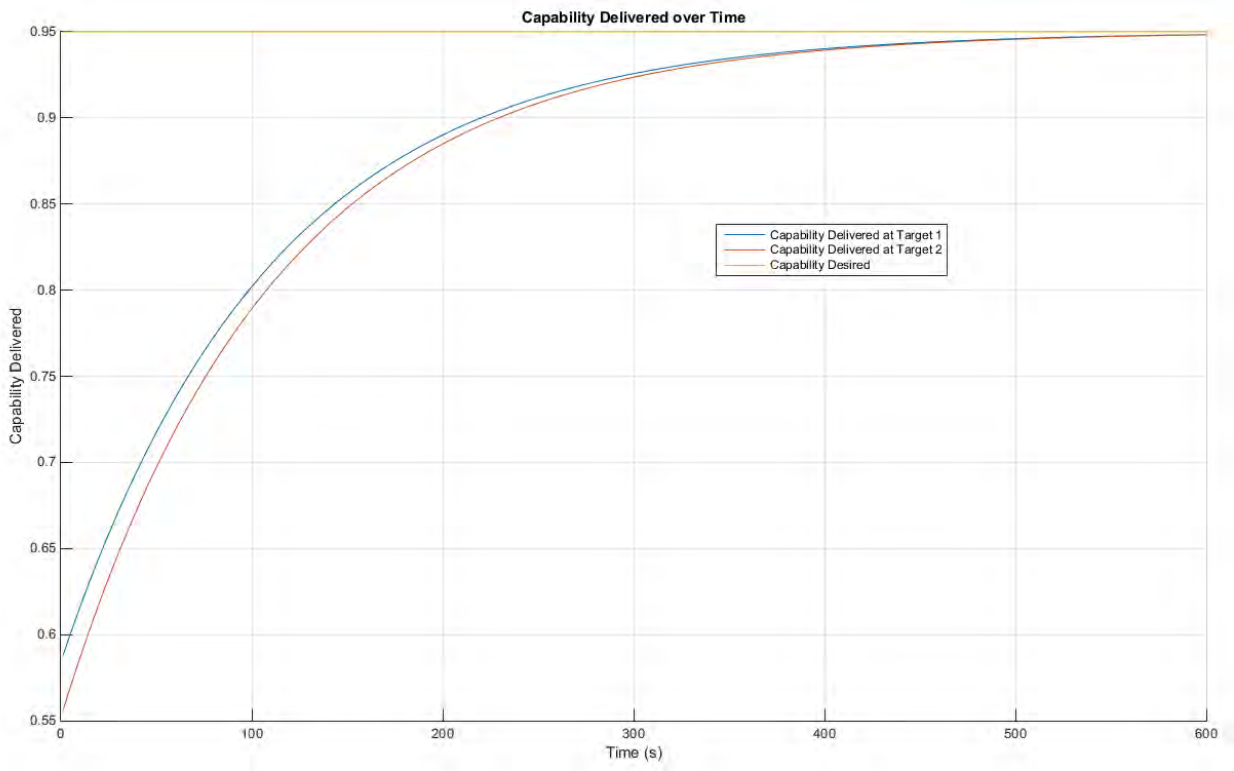

Figure 6.6: Change in Capability Over Time for Four Unit Swarm

While the swarm did manage to meet the desired capability as observed in Figure 6.6, it did so at the risk and peril of ignoring the obstacles while colliding with each other in the process. Due to the fact that the swarm units starting position were so close to each other, the change in the Jacobian pseudoinverse resulted in the swarm units following almost the same path as observed in Figure 6.3

6.3 Simulation with Hybrid Control

Now that the primary control has been shown to be working as intended, the next step was to incorporate the secondary control into the controller. This allows the swarm to have unit level intelligence and exhibit some level of behavioral control. Simulations were carried out for the exact same scenario as that of Figure 6.3 in order to contrast the difference between the controls. The result of the simulation is as seen in Figure 6.7.

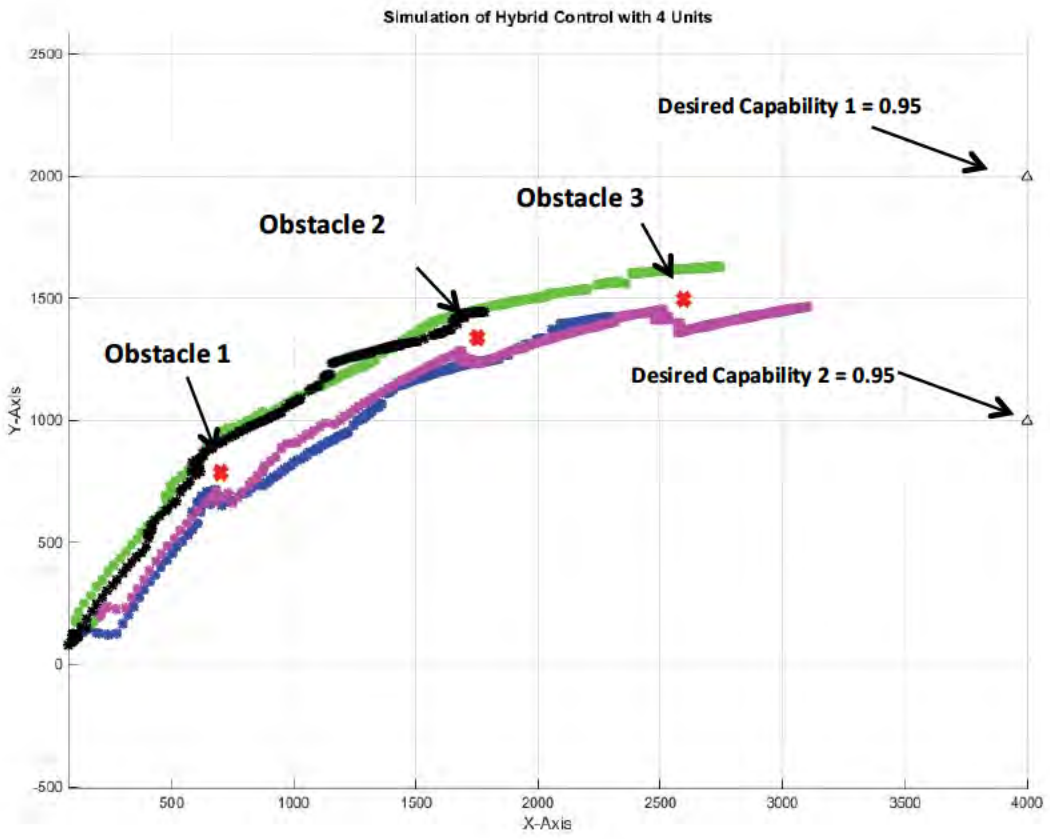

Figure 6.7: Four Unit Swarm with Two Targets for Hybrid Control

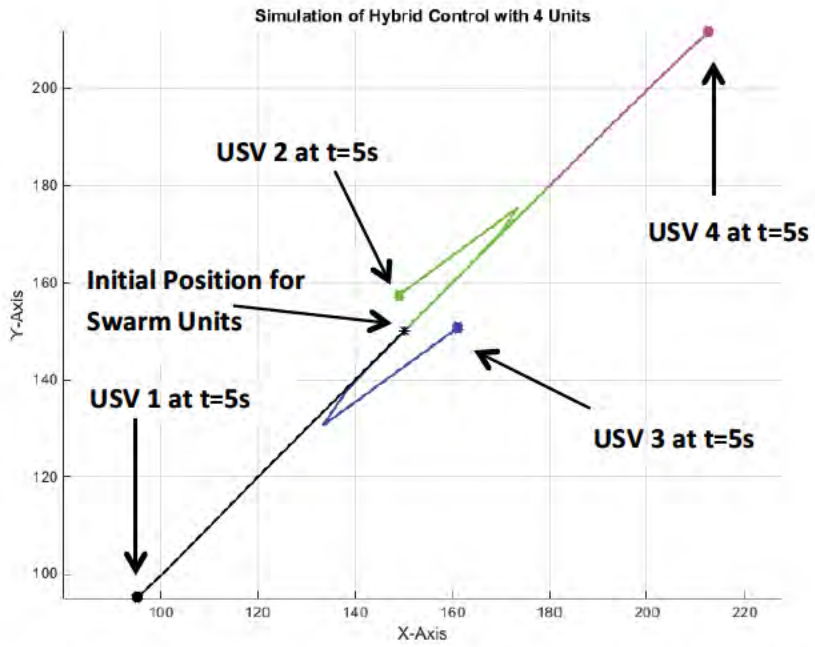

Figure 6.8: Four Unit Swarm with Two Targets for Hybrid Control at t = 5s

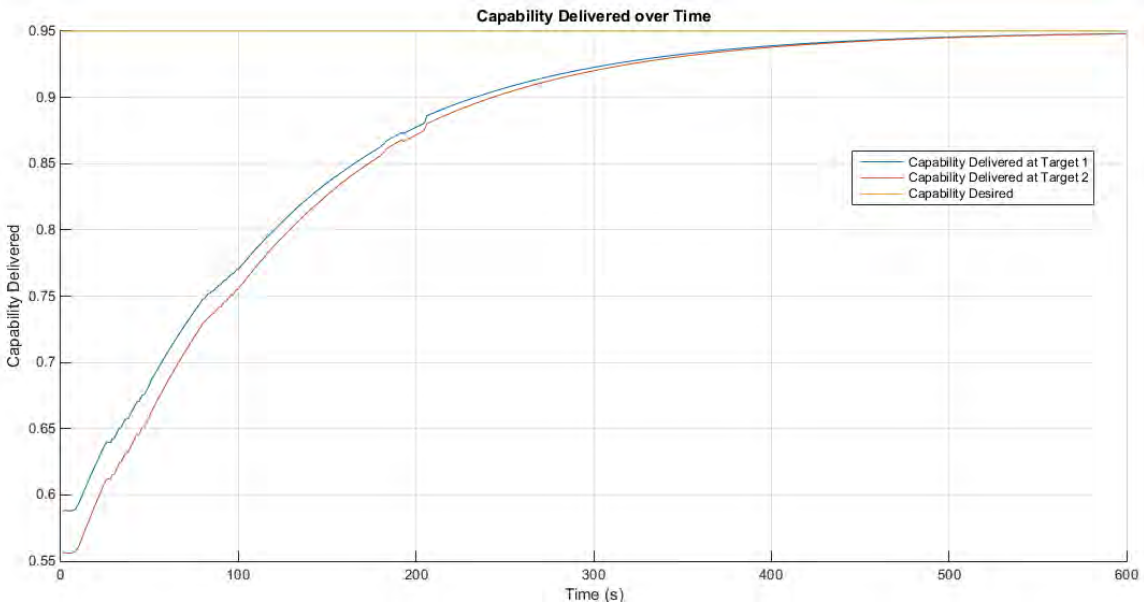

Figure 6.9: Capability Delivered Over Time for Hybrid Control

As observed in Figure 6.8, the immediate action for the swarm was to spread out such that they avoid any risk of collision. This adheres to one of the secondary objective laid out in Figure 5.1. Furthermore, the swarm units successfully navigated around the obstacles as observed in Figure 6.7. All secondary tasks were achieved without comprising the final capability delivered. However, as observed in Figure 6.9, the capability delivered was not as stable or smooth as that in Figure 6.6 due to the behavioral control introduced into the system, although this is merely a numerical artifact caused by the integration routine and very high gains.

CHAPTER 7 – SIMULATION FOR SUB-SURFACE

7.1 Overview

Thus far, the simulations have proved successful with regards to accommodating the secondary objectives while still achieving the primary objectives for USV systems. The simulations now shift to the sub-surface where there is an added dimension to consider. This complicates the coding process. However, the fundamentals of the control still remain the same.

Similar to the simulations done on the USVs, the UUVs undergo simulations to show that the secondary control indeed does have the desired effect on the system by contrasting the primary control with the hybrid control.

Furthermore, for the sub-surface simulations, the control gain for the primary control, K_{pri}, was set at 0.09 while the control gain for the secondary control, K_{sec}, was set at 45000. The *safezone* for obstacle and unit avoidance was set to 100 meters while the gain for the artificial potential field vector was set to 1200. The gains were obtained experimentally and based on observation on the swarm's performance. Furthermore, these gains were chosen in order to ensure that the movement of the swarm units be as realistic as possible.

7.2 Simulation with Primary Control

The simulation was first defined for a swarm of 3 UUVs, with their starting positions close to each other such that a collision would inevitably occur. Furthermore, 2 obstacles were set into the path of the UUVs. Utilizing just the primary control, the results of the simulation is as portrayed in Figure 7.1 and 7.2

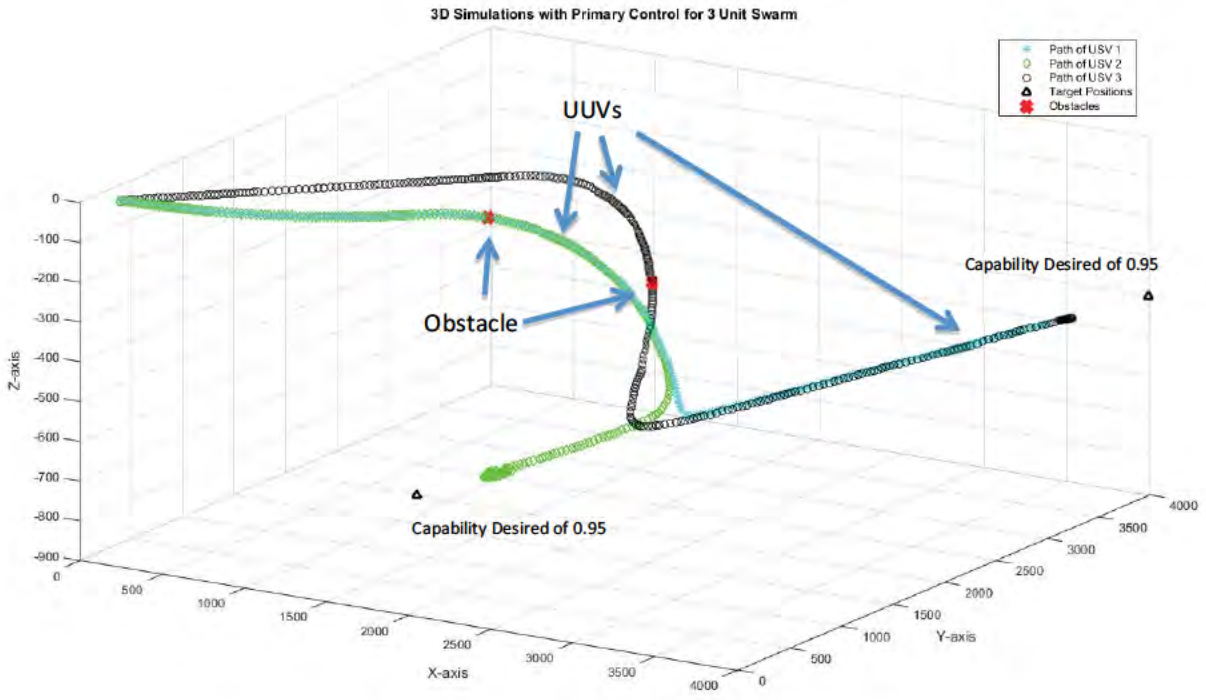

Figure 7.1: Front View for Sub-Surface Primary Control of 3 UUVs

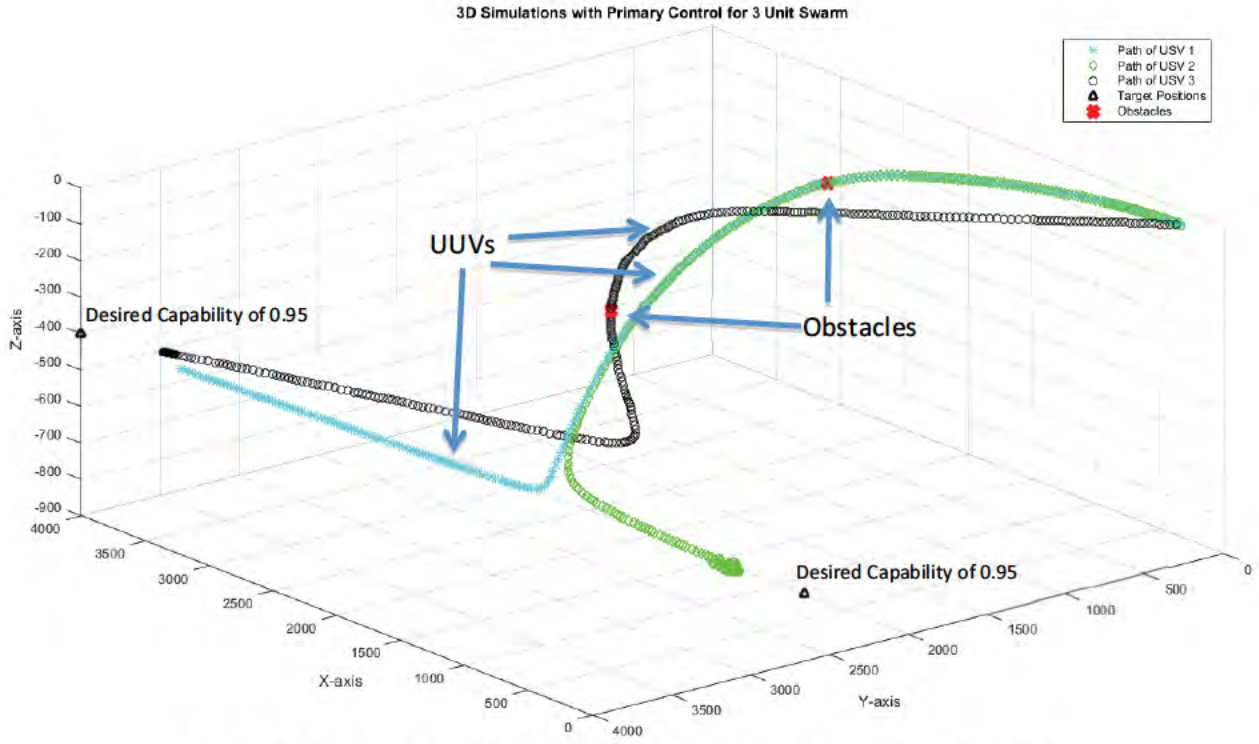

Figure 7.2: Back View for Sub-Surface Primary Control of 3 UUVs

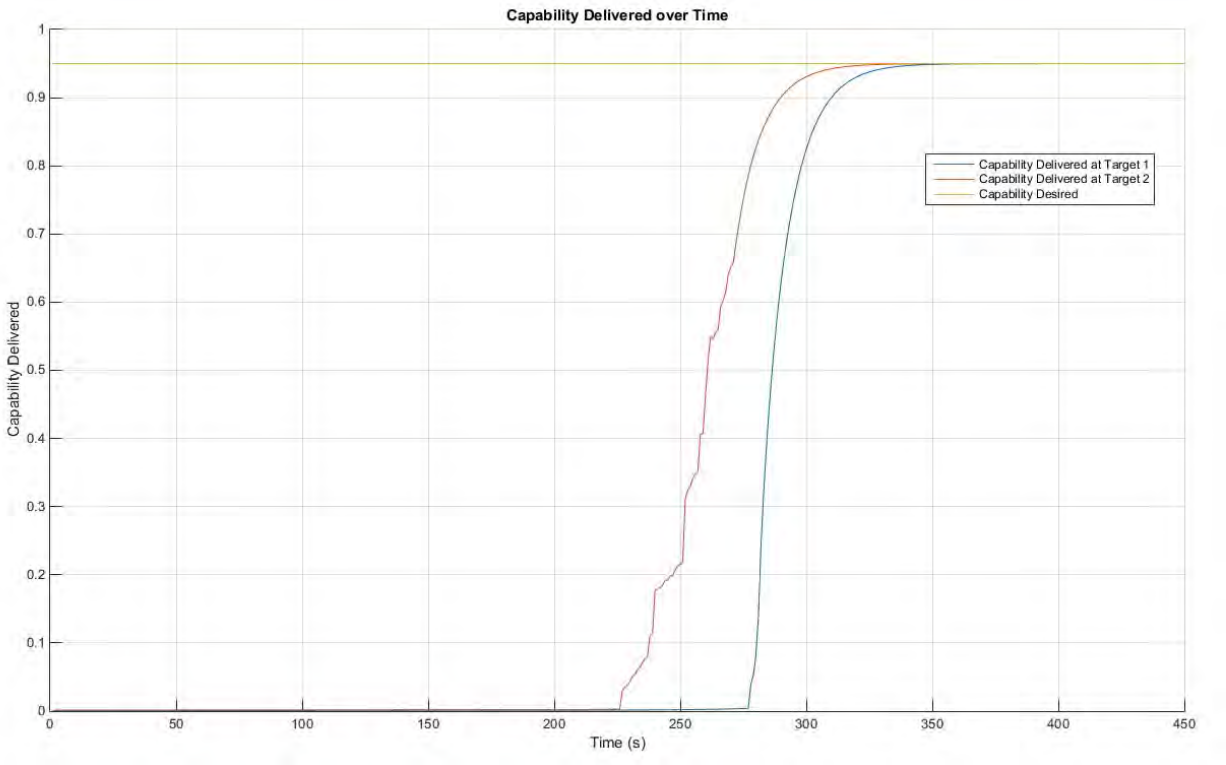

Figure 7.3: Capability Delivered over Time for Sub-Surface Primary Control of 3 UUVs

As can be seen in Figure 7.1 and 7.2, the swarm initially started with all units at the same starting position, but made no attempt to spread out in order to avoid collision. Furthermore, UUV 2 and UUV 3 mostly followed the same path in order to deliver the required capability. The UUVs also made no attempt to avoid the obstacles. However, at the end of the simulation, the capability delivered was as desired. Therefore, the primary control has effectively performed the objective defined for that aspect of the controller.

7.3 Simulation with Hybrid Control

Incorporating the secondary control into the system, we expect the UUVs to achieve the secondary objectives in Figure 5.1. The simulations results are as shown in Figure 7.4 and 7.5.

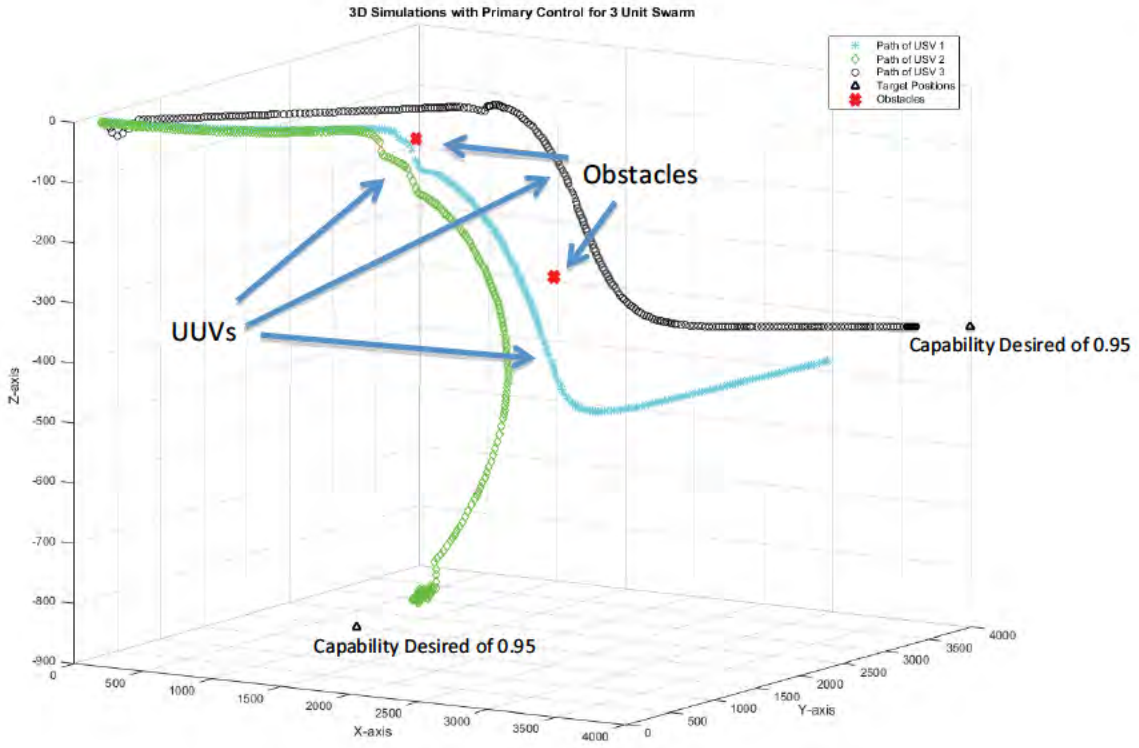

Figure 7.4: Front View for Sub-Surface Hybrid Control of 3 UUVs

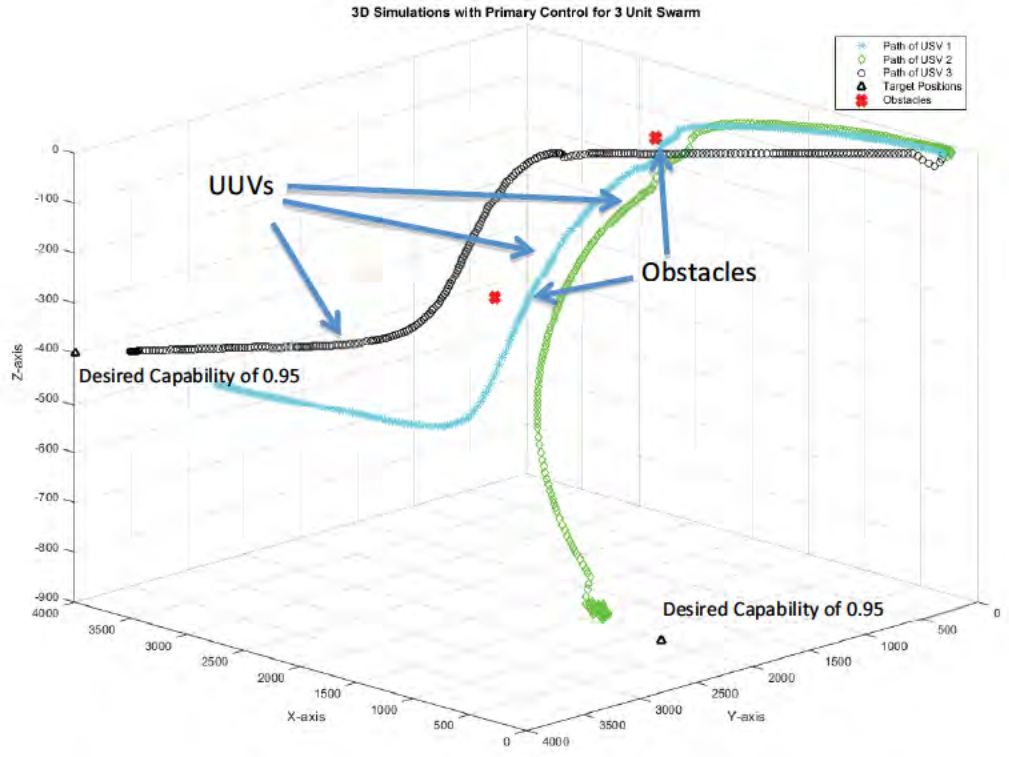

Figure 7.5: Back View for Sub-Surface Hybrid Control of 3 UUVs

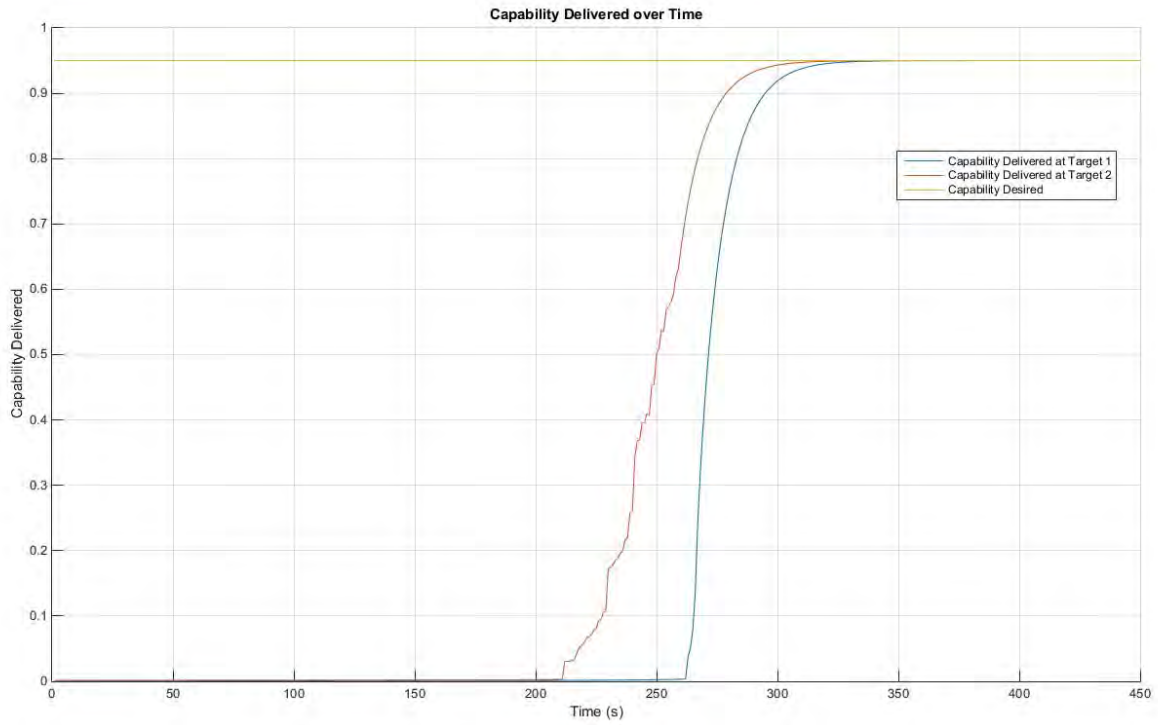

Figure 7.6: Capability Delivered over Time for Sub-Surface Hybrid Control of 3 UUV

As can be seen in Figure 7.5 and 7.6, the swarm's secondary control enabled the swarm units to avoid each other as well as the obstacles. Most notably, UUV 2 and UUV 3 did not follow the same path but instead actively tried to keep a distance of 100 meters away from each other due to a repulsive vector generated from APF theory as mentioned Chapter 5.3. In Figure 7.6, capability desired was achieved in the relatively same time frame as the primary control. Hence, the secondary control not only fulfills the secondary objectives but is effective as well.

CHAPTER 8 – COMBINED SIMULATION

8.1 Overview

The simulations now combine the USVs with the UUVs in order to provide some cooperative element to the hybrid swarm. Two hybrid controllers were defined separately for the USVs and the UUVs. However, included in the secondary control was an avoidance behavior based on APF for the USVs and the UUVs if they strayed too close to each other. Therefore, this system assumes that the position of the UUVs and USVs are known to each other via measurement using the primary sensors as well as explicit communications. This ties in further into cooperative control which will be explored in later portions of the project.

Both the primary controller and then the hybrid controller were implemented separately with both USVs and UUVs in order to contrast the difference between the two and demonstrate the effectiveness of the hybrid control. Furthermore, the swarm environment utilized in these simulations differs from the simulations of Chapter 6 and Chapter 7 in order to show the adaptive and flexible nature of the hybrid control in dealing with an unknown environment.

The gains used in this simulation were identical to the gains used as mentioned in chapter 6.1 and 7.1.

8.2 Simulation with Primary Control

In this simulation, 3 UUVs and 2 USVs with two positions where capabilities are desired is portrayed in Figure 8.1. Furthermore, obstacles were placed around the environment, and the USVs and UUVs started relatively close to each other in order to contrast the effect of the hybrid control in fulfilling the secondary objectives. The result of the simulation utilizing only the primary control is as follow in Figure 8.1 and 8.2.

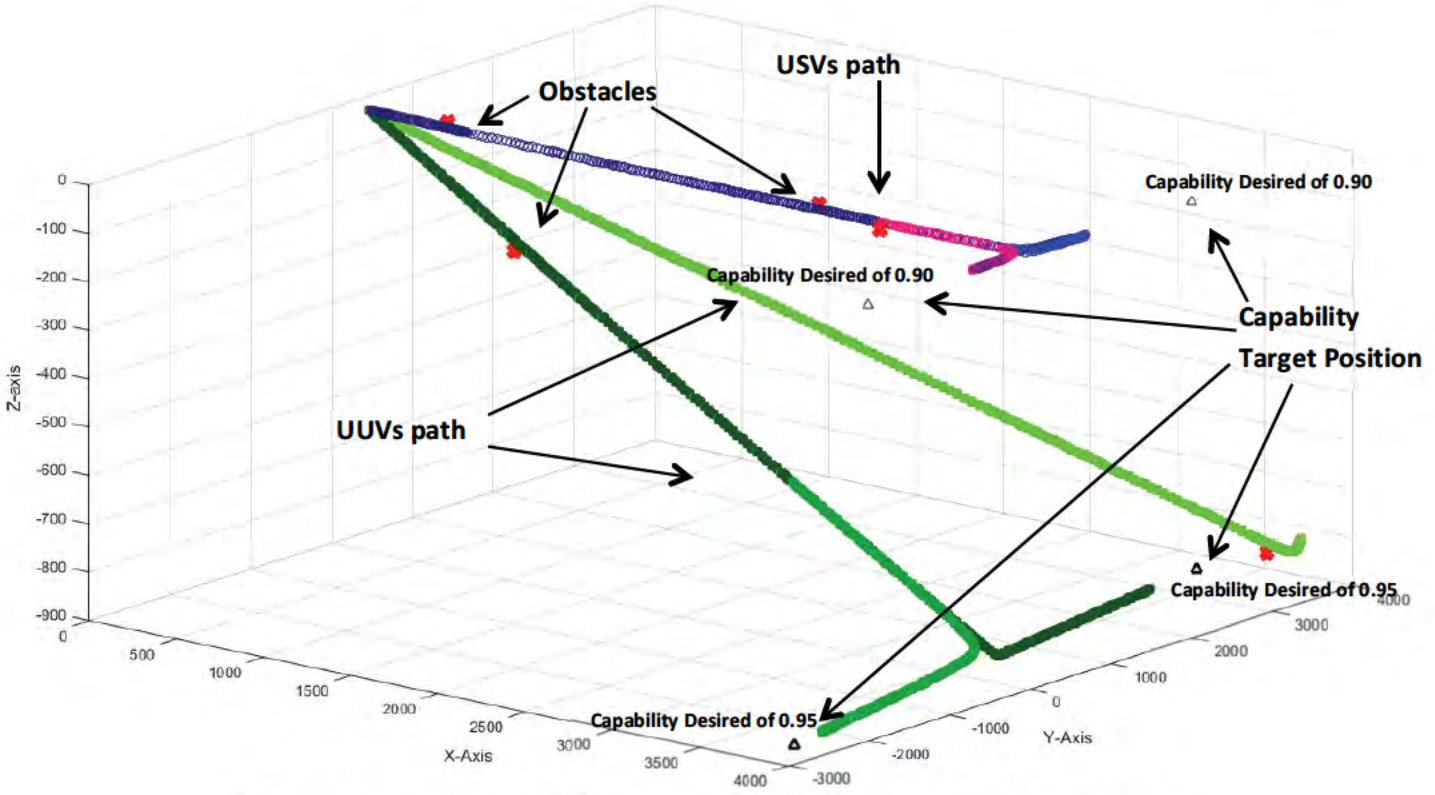

Figure 8.1: Front View for Combined Hybrid Control of 4 USVs and 3 UUVs

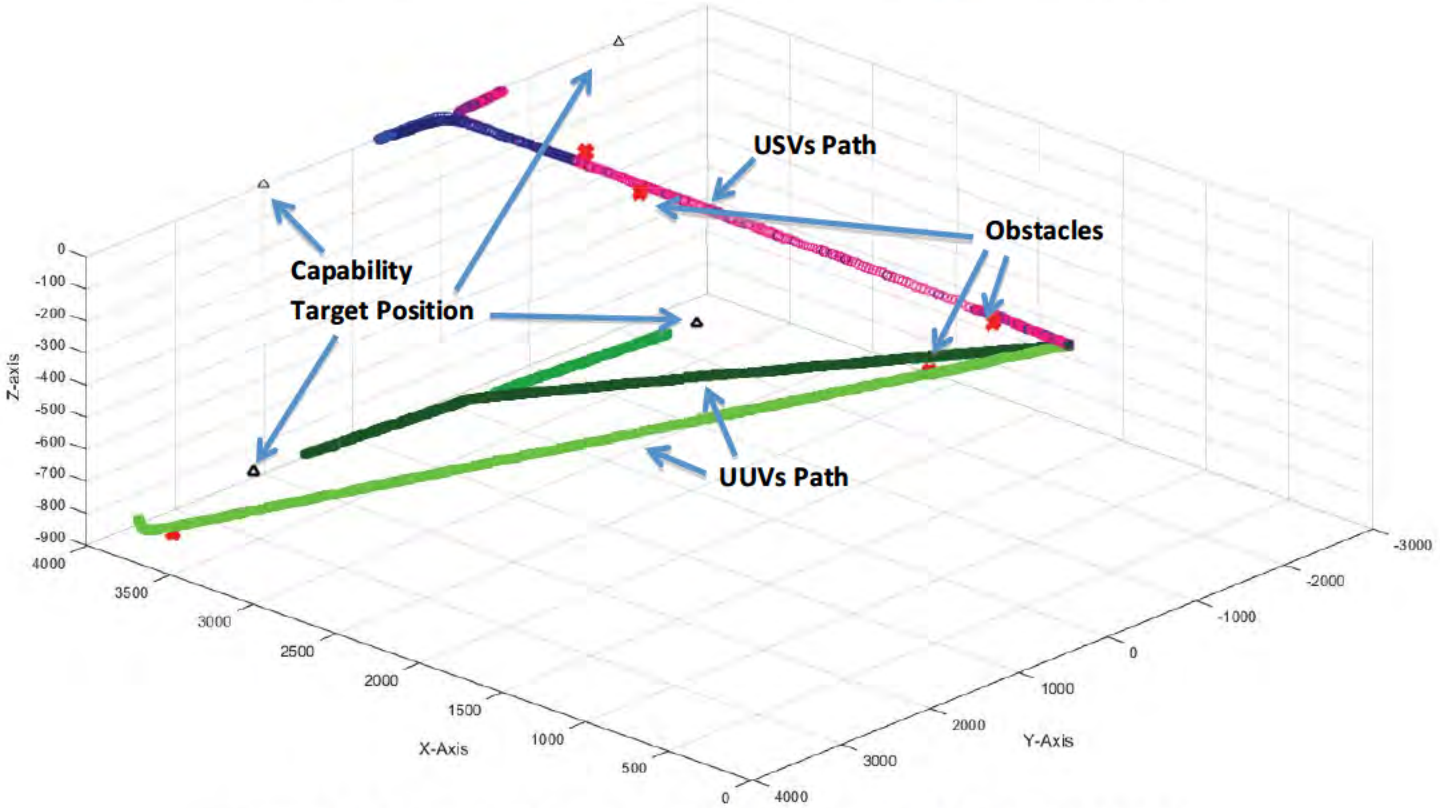

Figure 8.2: Back View for Combined Hybrid Control of 4 USVs and 3 UUVs

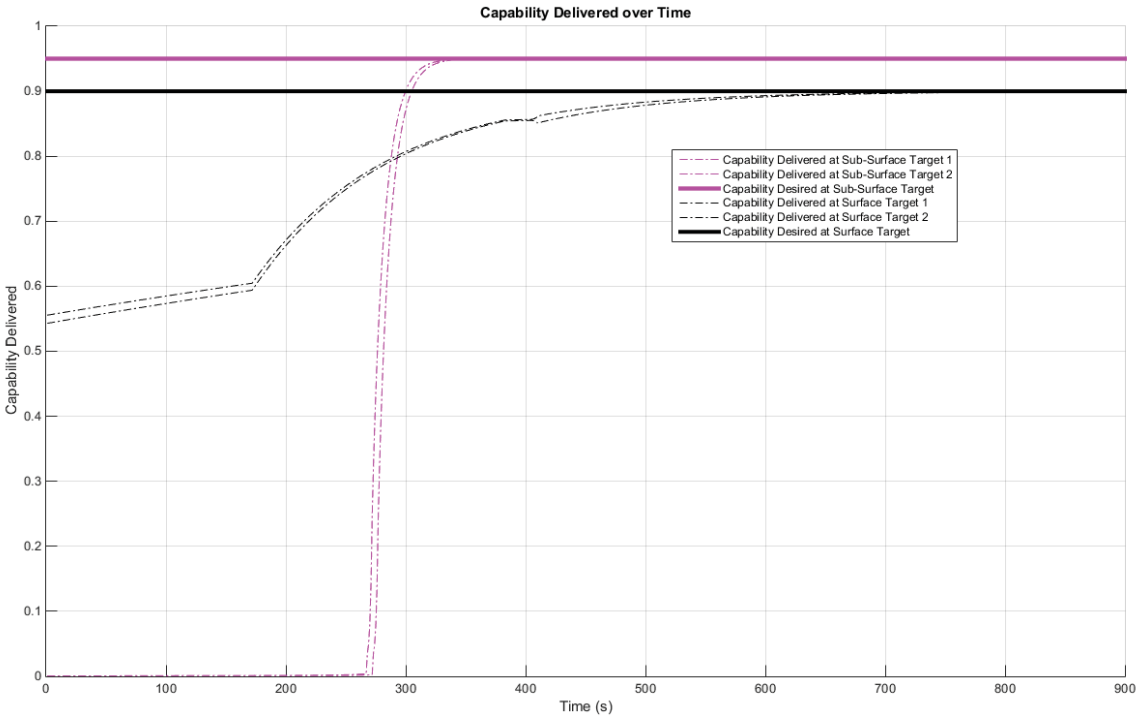

Figure 8.3: Capability Delivered over Time for 4 USVs and 3 UUVs

As can be seen, the capability desired was met and delivered by the swarm. However, the swarm made no effort to avoid obstacles or separate the units to avoid wasted capability and collisions. Furthermore, some of the USVs and UUVs followed the same path in order to deliver the capability. In order to achieve the secondary objective, the hybrid controller was implemented for the full swarm, as seen in the next section.

8.3 Simulation with Hybrid Control

In the hybrid control, the simulation environment was set to be the same as that in the primary control in order to contrast the difference between the two controllers. Most significantly in this simulation, the USVs and UUVs must move such that they take into account each other's position and actively attempt to avoid each other. The secondary objectives in this swarm are similar to the objectives as shown in Figure 5.1. Figure 8.4 to Figure 8.7 shows the result of the simulation with the hybrid controller.

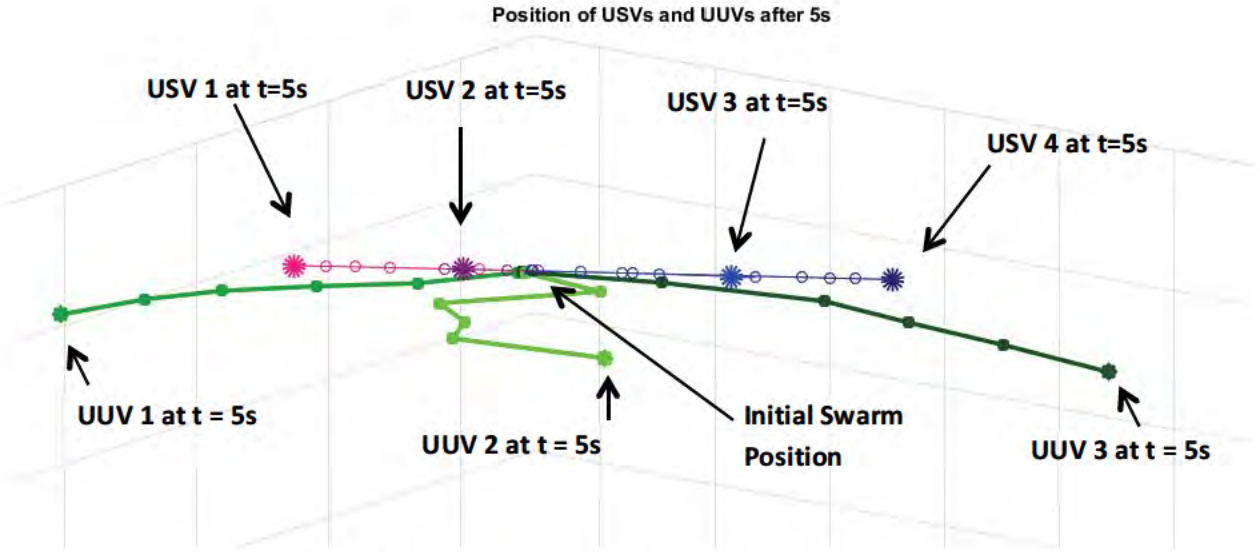

Figure 8.4: Position of USVs and UUVs after T = 5s

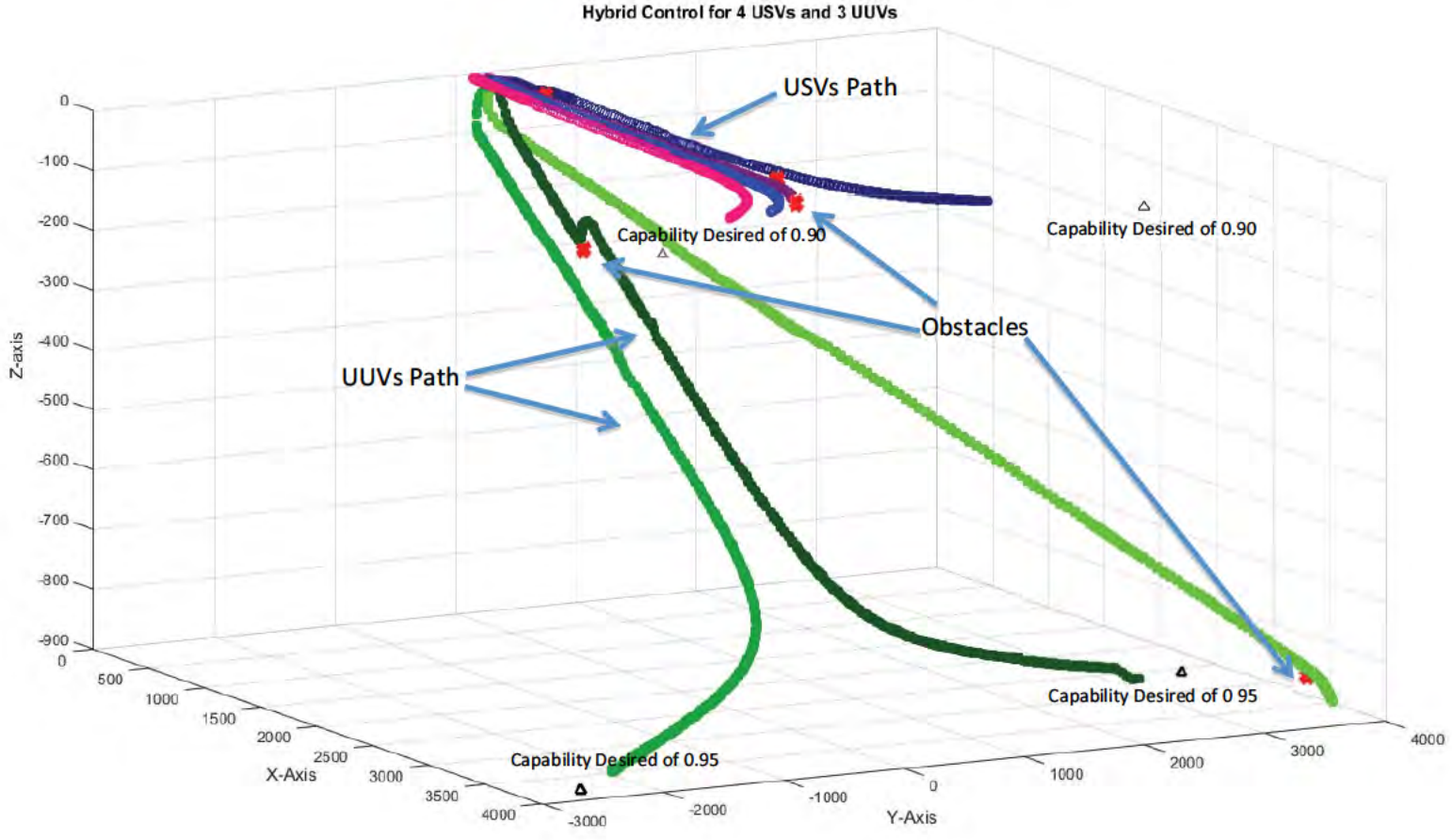

Figure 8.5: Front View for Hybrid Control of 4 USVs and 3 UUVs

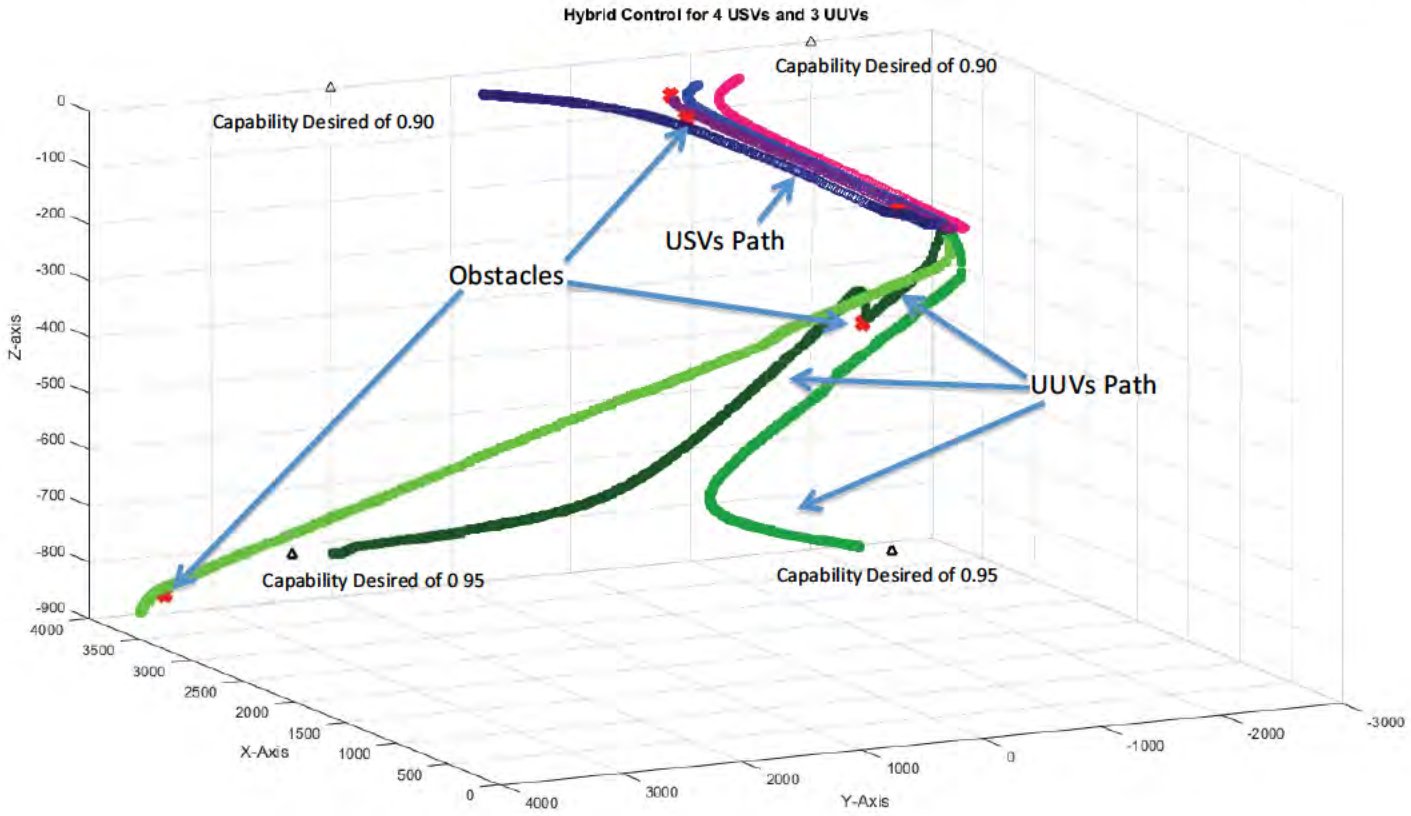

Figure 8.6: Back View for Hybrid Control of 4 USVs and 3 UUVs

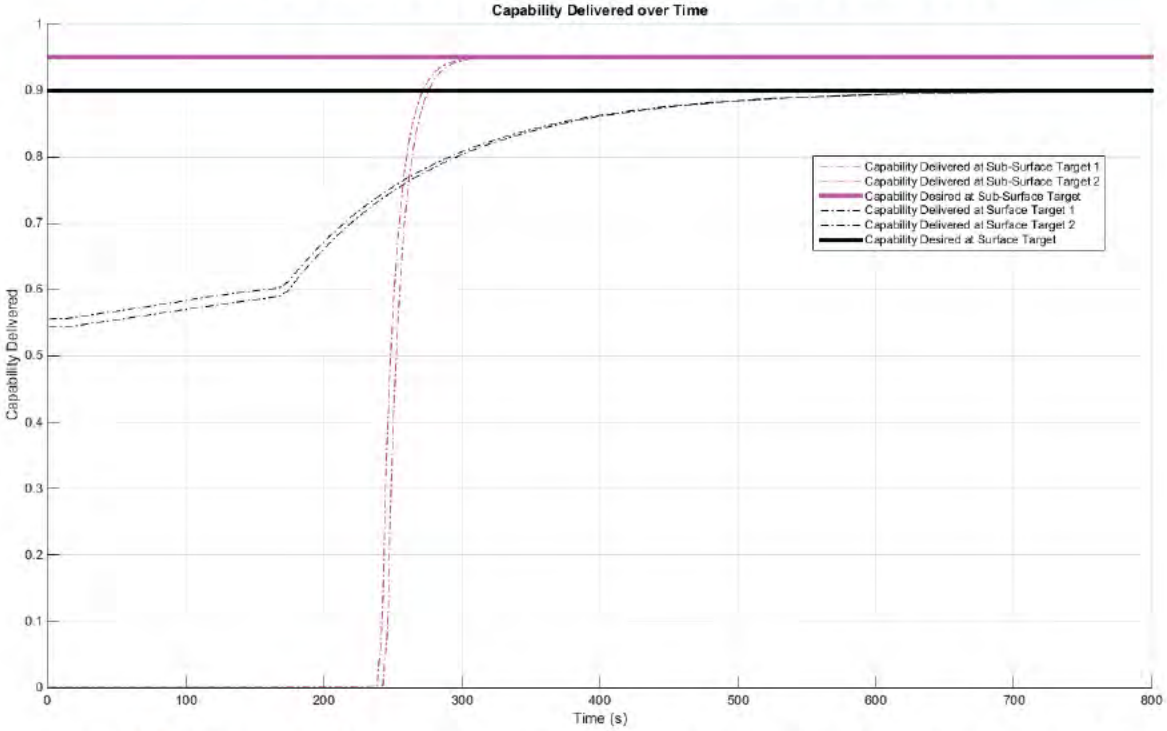

Figure 8.7: Capability Delivered over Time for Hybrid Control of 4 USVs and 3 UUVs

As seen in Figure 8.4, through the secondary behavioral control, the swarm units alternated their speed such that they spread out apart from each other in order to prevent collision. Furthermore, with the hybrid control introduced into the system, the swarm units maintained a certain safe distance apart from each other while ensuring that they avoided the obstacles that were placed in their path as seen in Figure 8.5 and Figure 8.6. Finally, the secondary control did not affect the primary task, which was to deliver the required capability as seen in Figure 8.7. Overall, the hybrid control proves successful for a swarm of USVs and UUVs working cooperatively together.

CHAPTER 9 – COOPERATIVE LOCALIZATION OF UUVS

9.1 Overview

Localization of vehicles beneath the surface presents a very challenging problem given that active localization in the form of Global Position Systems (GPS) is not available in the sub-surface domain. As such, UUVs operating on their own typically rely on inertial navigation systems (INS) based on accelerometers and rate gyros to give the system a sense of where the unit is located in some reference frame. However, there is an innate problem that is typically associated with INS – even the best INS has noise-induced error that compounds over time. This is due to the lack of external reference, as the system is effectively "flying blind" under the water. It is estimated that typical navigation errors range from 0.5% to 2% of distance traveled, even when the UUV is operating with a Doppler Velocity Log [20] that can provide reference to the environment as motion occurs.

The traditional form of active localization that is typically utilized by UUVs is achieved via static beacons that are able to communicate with the units in order to provide navigational and positional information. This technique, usually called long baseline (LBL) localization, has limitations in that it severely limits the area of operations of the UUVs, restricting them to operate near the beacons. Furthermore, this requires infrastructure preparation in the theater of operations, which may not always be feasible. However, LBL does provide the basic concept of how any form of active localization underwater would work – that is, through a third party providing positional and navigational data. Figure 9.1 shows the basic workings of the long baseline technique where a submersible vessel is working with static beacons. These static beacons transmit positional data based on their own GPS measured or *a priori* known position in order to produce a "fix" on the submersible vehicle.

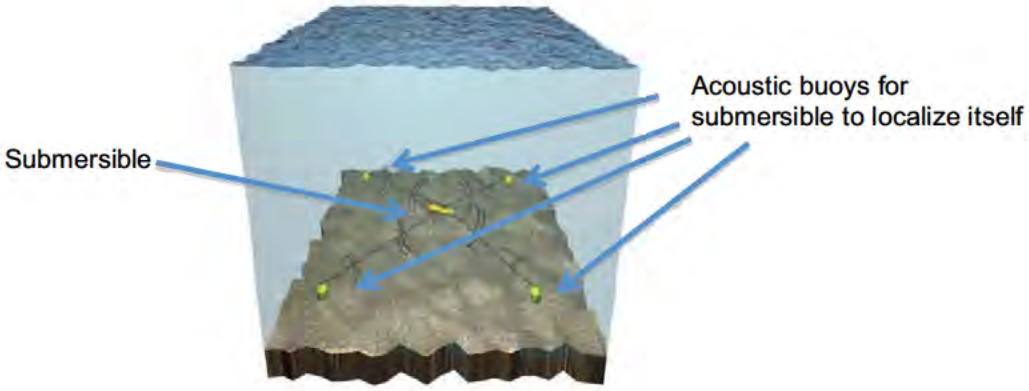

Figure 9.1: Submersible Utilizing LBL System [21]

9.2 Previous and Current Works

In lieu of utilizing beacons as navigational aids, a team at MIT has developed an algorithm that allows surface vessels to be used as navigational aids to localize UUVs [20]. Termed as cooperative navigation (CN), the algorithm uses dead-reckoned positional information from the USV forward-propagate in time to the control. This combined with the latest range measurement between the USV and UUV would provide a fix that allows the system to pinpoint the position of the UUVs. Figure 9.2 shows a diagram of the CN-algorithm.

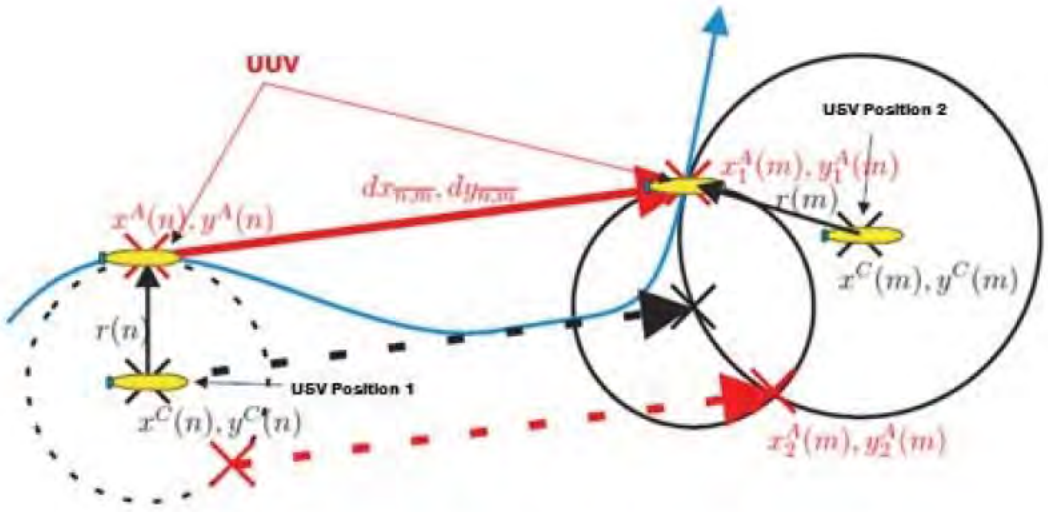

Figure 9.2: CN-Algorithm to Calculate Position of UUVs [20]

The actual positions of the UUV at time *n* and *m* are as pointed out by the red arrows. At time *n*, the USV would initially transmit range data to the UUV given out by *r(n)*. Using the dead-reckoning information given by $d_{x_{\overline{n,m}}}, d_{y_{\overline{n,m}}}$, combined with the range data from ASV 2 at time *m,* given by *r(m)*, the algorithm would then be able to localize the position of the UUV.

Furthermore, actual experiments utilizing this CN-algorithm have been done by MIT and the results have proved successful thus far. Using a "Scout" autonomous surface craft which they termed as a cooperative and navigation aid (CNA) and a Bluefin UUV, the team carried out the experiment utilizing the CN-algorithm. The results are as shown in Figure 9.3.

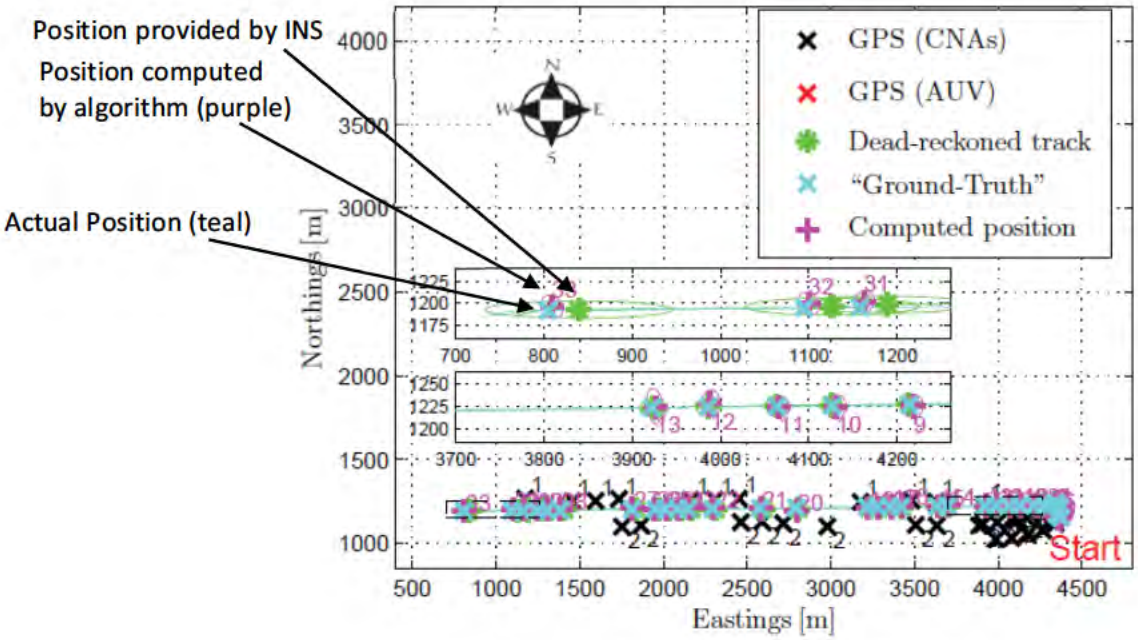

Figure 9.3: Results of CN Algorithm [20]

Effectively, the experiment had the UUV run in a 4km straight track from east to west. The computed location was then compared with the dead-reckoned position and the actual position. Initially, the "dead-reckoned" position is close to the ground truth and computed position. However, over time, the INS position estimate degrades, and therefore the dead-reckoned position strays further away from the actual location of the UUV. However, the

computed position still remains relatively accurate. This shows the effectiveness of this CN-algorithm in localizing the UUVs. Moving forward, this CN-system is to be implemented as a secondary objective into the developed swarm controller.

9.3 Implementation of Cooperative Navigation

The UUVs were assumed to have an INS drift of 1.6% for every second of travel underwater. This drift value is dependent on the exact model of INS on the UUV, and can be adjusted accordingly for any system. The important realization is that the drift may result in inaccurate positioning of the UUV units and therefore result in a degraded capability, as can be seen in Figure 9.4.

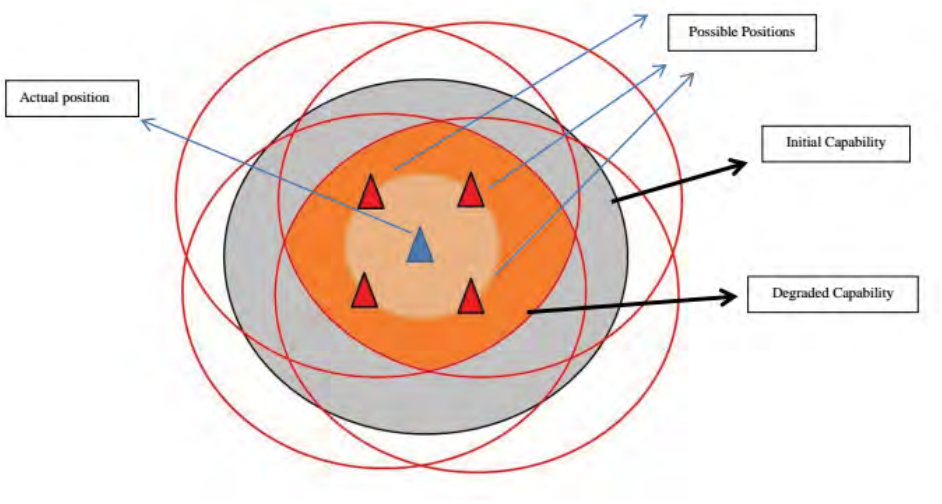

Figure 9.4: Capability Degradation of UUVs

The initial capability provided by the UUV is given by the grey circle. However, due to the drift associated with the INS, several possible positions of the UUV are given by the different red triangles. As there is no certainty as to the position of the UUV, the worst case scenario must be assumed and therefore, the final capability provided to the system is given by the orange circle.

Utilizing the current control, the most logical method in implementing CN was to make use of the secondary control to manipulate the UUVs to find a USV in order to get localized when needed. Furthermore, it is important that in this process that the capability provided to the system remains at its most optimal. This is done by adding in an attractive vector that would pull the UUVs to the USV to get localized when a certain threshold is met, but otherwise to allow them to move as dictated by the standard hybrid controller.

As this paper mainly concerns itself with the control of the USVs and UUVs, the control does not concern itself with the actual intricacies of the CN-algorithm. Furthermore, we simply assume that localization happens when the UUVs come within a 500m radius of the USVs, which is the maximum range of the acoustic modem. As for capability degradation, utilizing the INS drift of 1.6%, the worst case position of the UUVs were assumed in order to calculate the current capability provided to the system.

9.4 Simulation without Cooperative Navigation

The first simulation was done without utilizing CN in the system. Instead, an INS drift based as a function of time was assumed and no mitigation was provided. Over time, the capability of the UUVs degraded. Furthermore, the control system would take into account the drift-based degradation of capability and correct unit positions until desired capability was deemed to have been provided to the system under the worst-case scenario. Figure 9.5 shows the environment that the simulation was carried out in and Figure 9.6 shows the end state of the simulation.

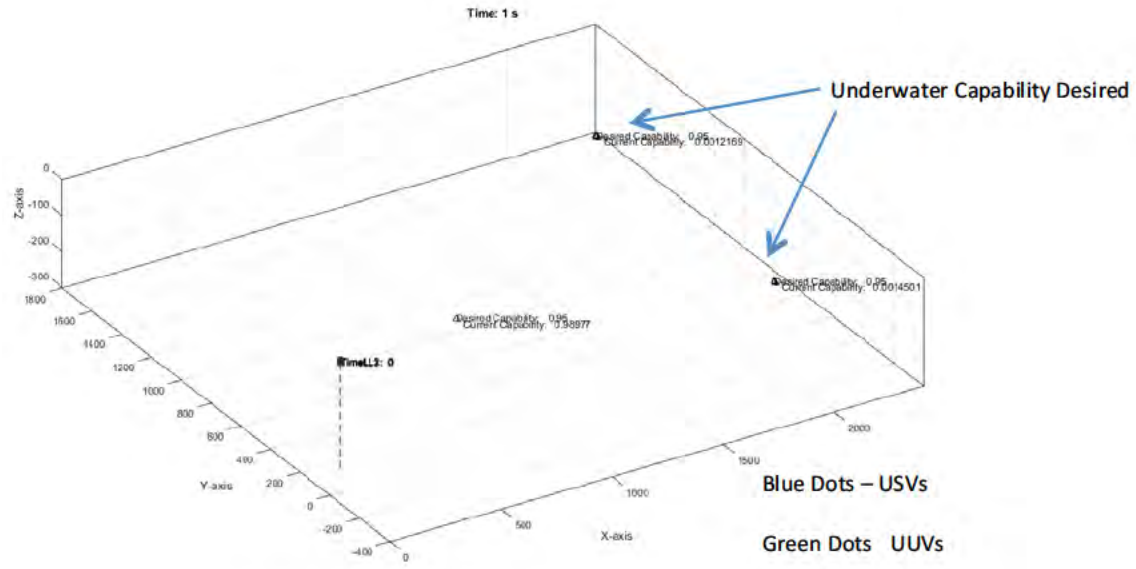

Figure 9.5: Initial Simulation Set-up

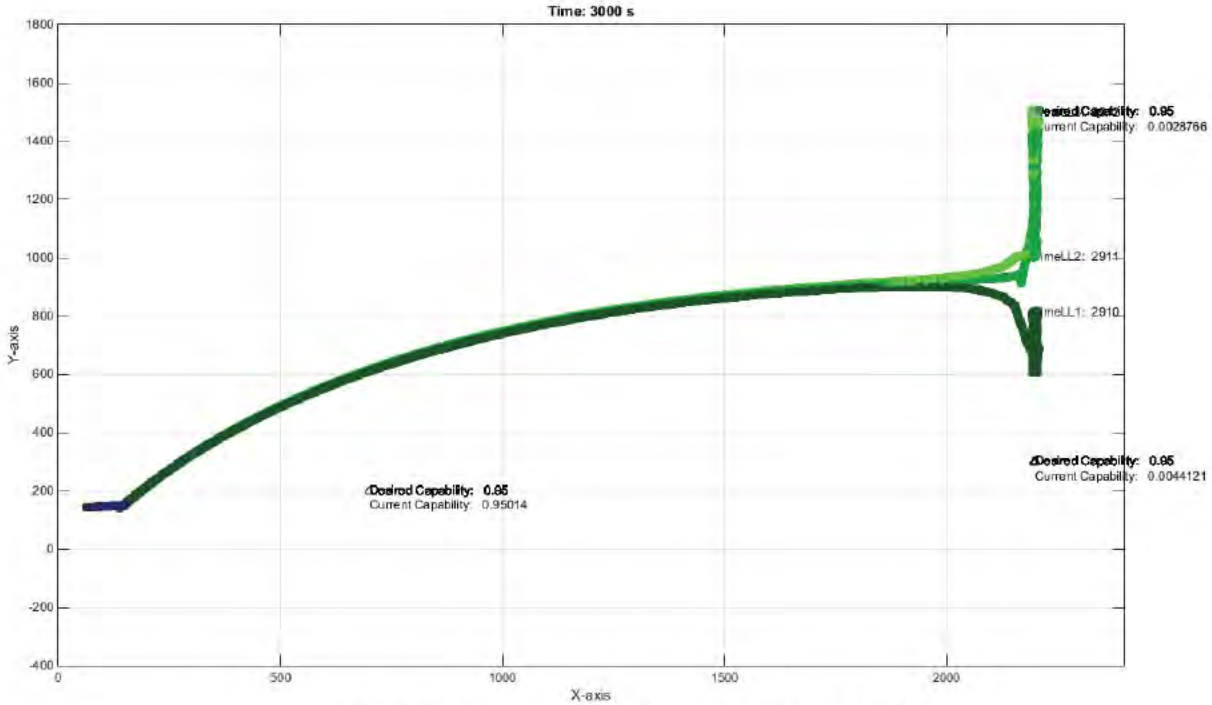

Figure 9.6: Simulation End-State (Top View)

The control system attempts to move the *perceived* position of the swarm UUV units such that the desired capability is achieved even when there is localization error, using the

worst-case scenario. That is, the controller utilizes the estimated position of the unit in the control, but computes the capability as if the unit were in the worst possible position, based on the estimated uncertainty. Eventually, the capability degrades to such an extent that the control system simply attempts to place the UUV right on the position where capability is desired. At this position, the system provides the maximum available *degraded* capability and therefore, the resulting capability provided to the system is merely a function of the INS drift. That is, the controller attempts to place the unit on the target, but recognizes that the actual position may be far away and computes delivered capability based on that potential position. Eventually, that degraded capability drops below the desired value. Figure 9.7 shows the relation between time and the capability provided. The noise is caused by the localization error and uncertainty mitigation.

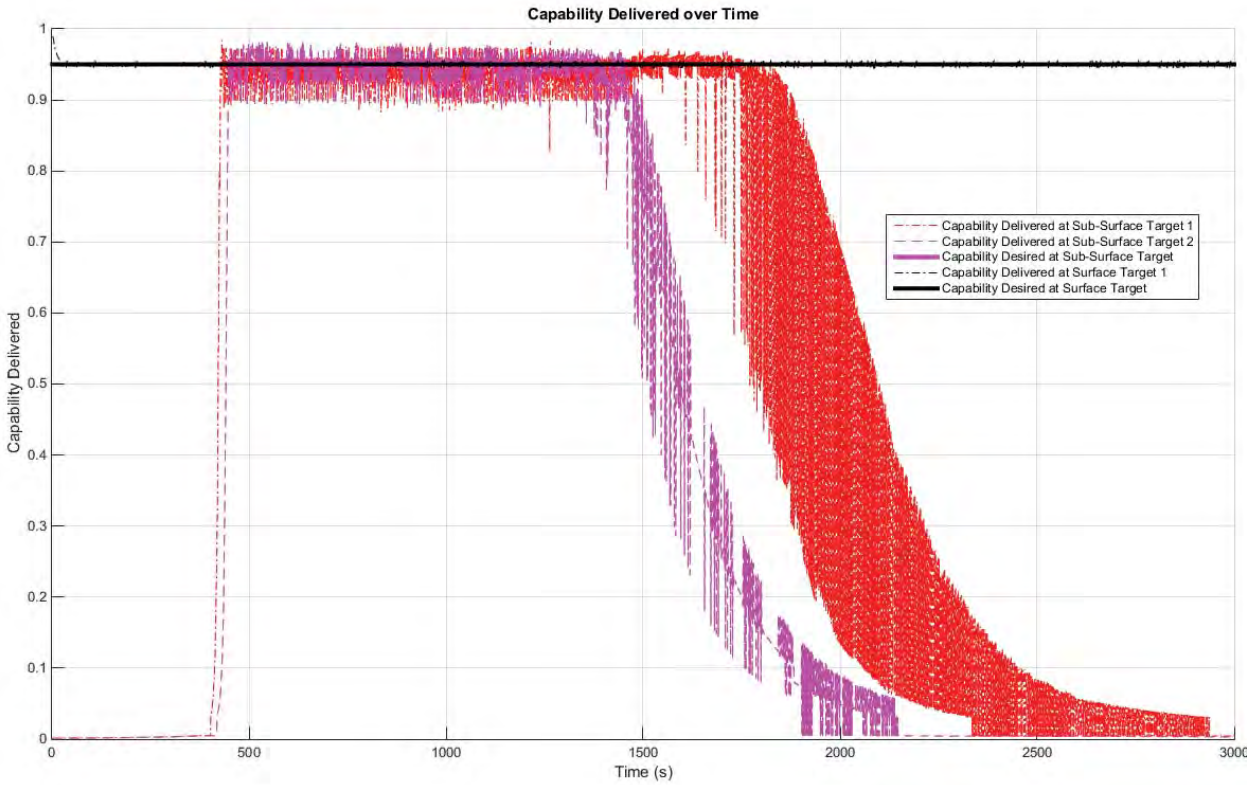

Figure 9.7: Capability Delivered Over Time without CN

9.4 Simulation with Cooperative Navigation

Simulations were then done with the attractive vector that would simulate the CN capability embedded into the secondary controller. After a UUV navigates unaided for 300 seconds or more, it experiences an attractive force through the secondary control that pulls it toward the closest USV to get localized. Upon achieving a localization update, the attractive potential is removed for that unit. Figure 9.8 to 9.12 shows the results of the simulations in a snapshot format. The figures are provided via a top-down view of the theater of operations to provide easier viewing since the point of this simulation is to ensure that the CN-algorithm would work. The blue dot represents the USV while the green dots are representative of the UUVs.

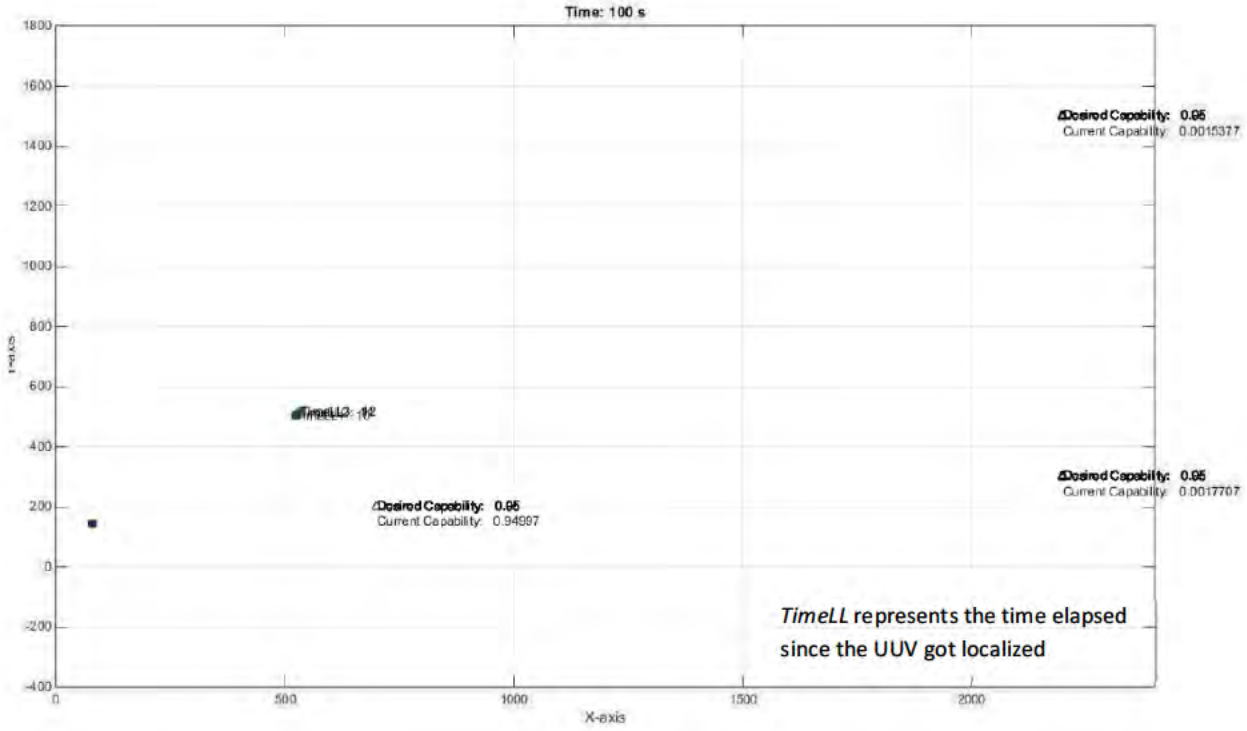

Figure 9.8: Initial Simulation

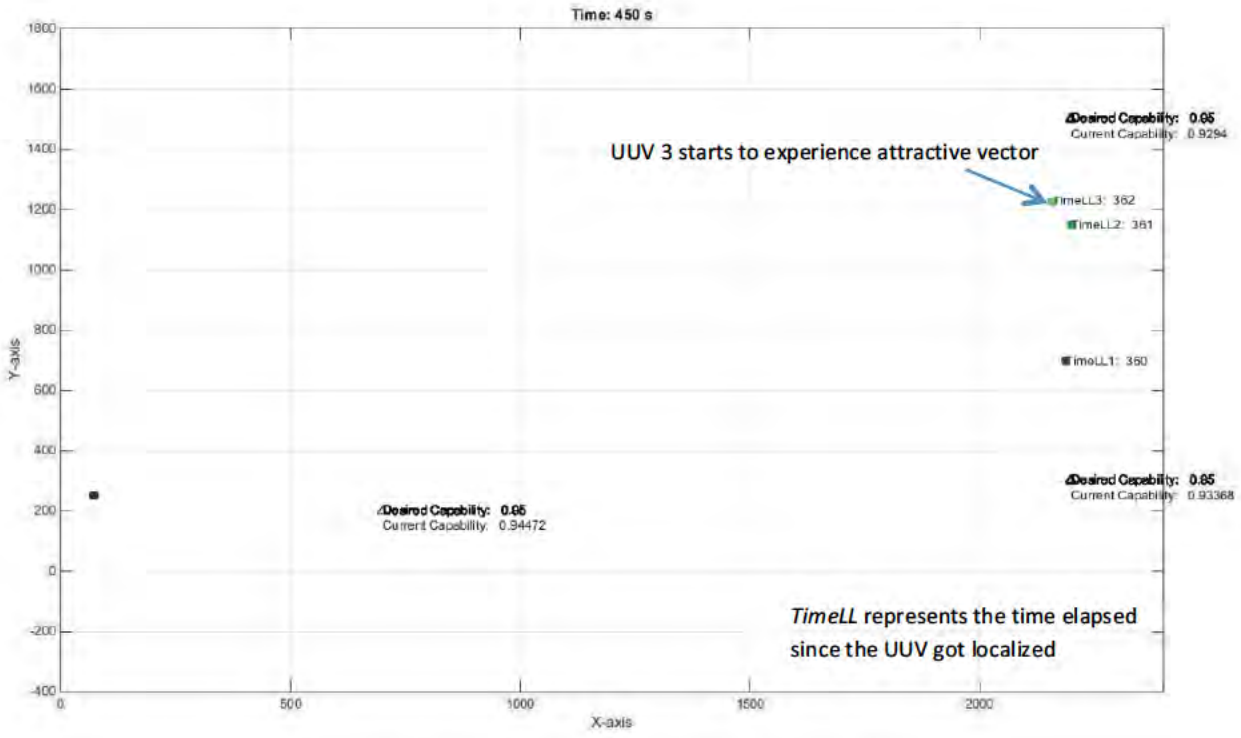

Figure 9.9: UUVs Providing Required Capability

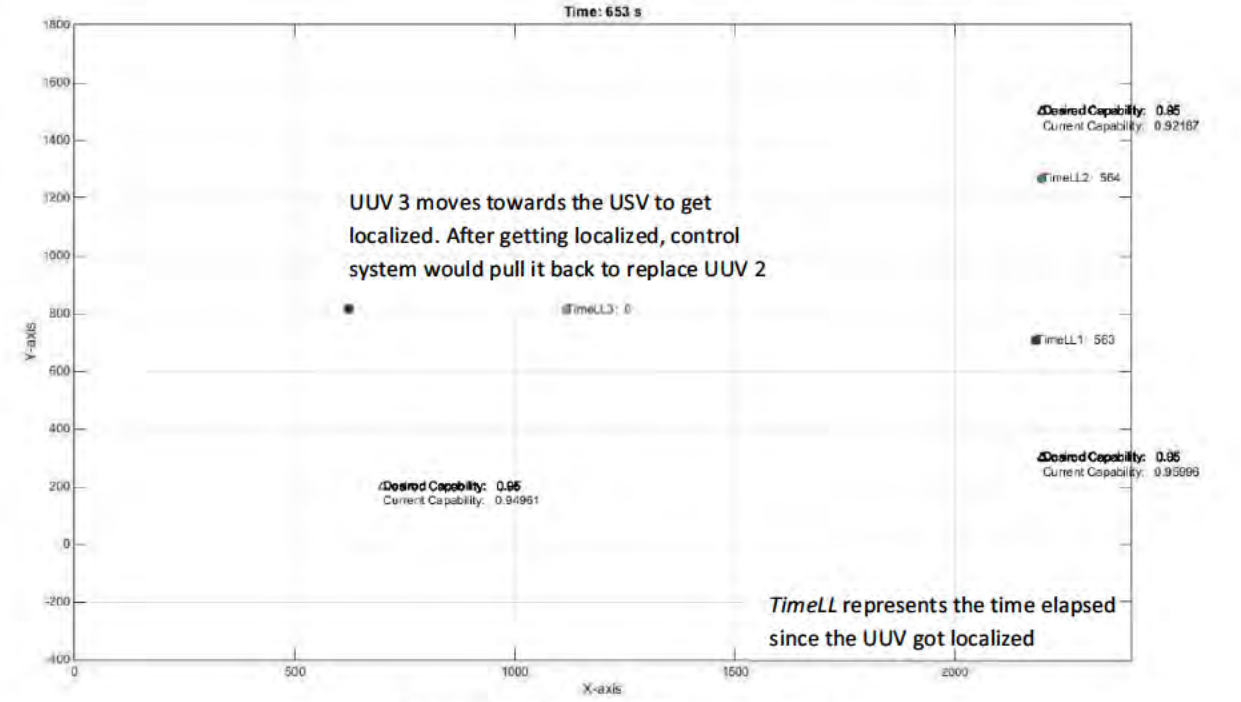

Figure 9.10: UUV 3 Gets Localized

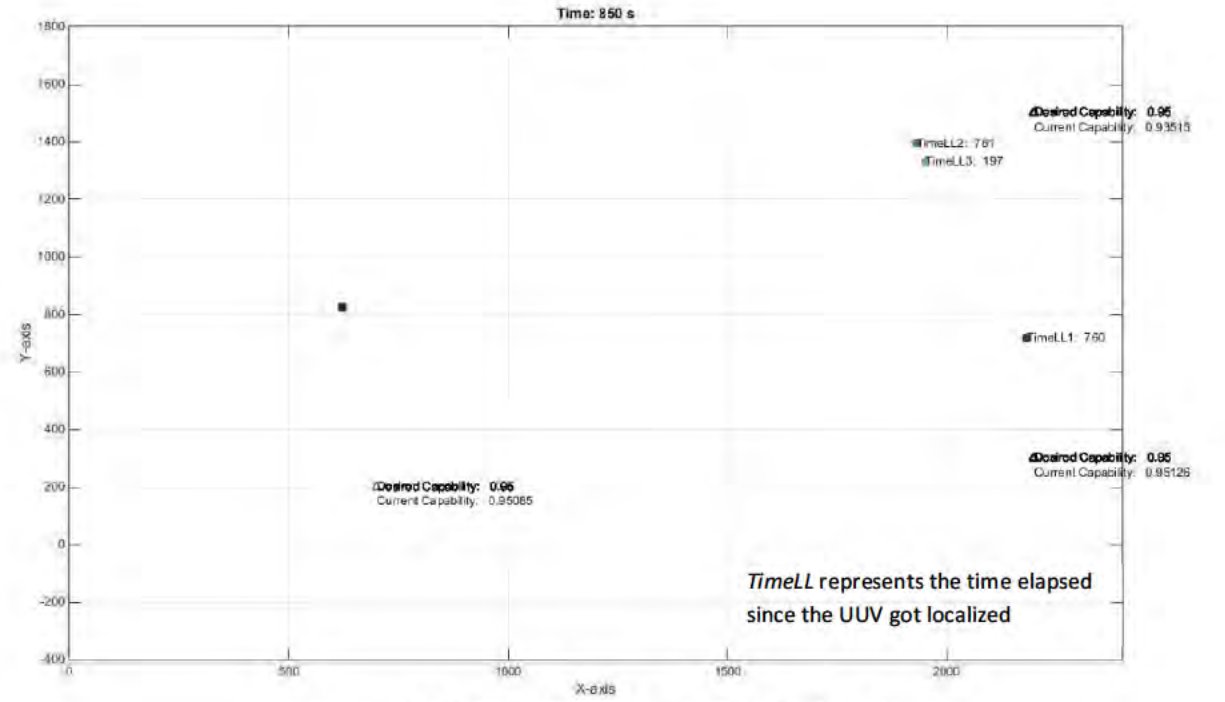

Figure 9.11: UUV 3 Replaces UUV 2's Position

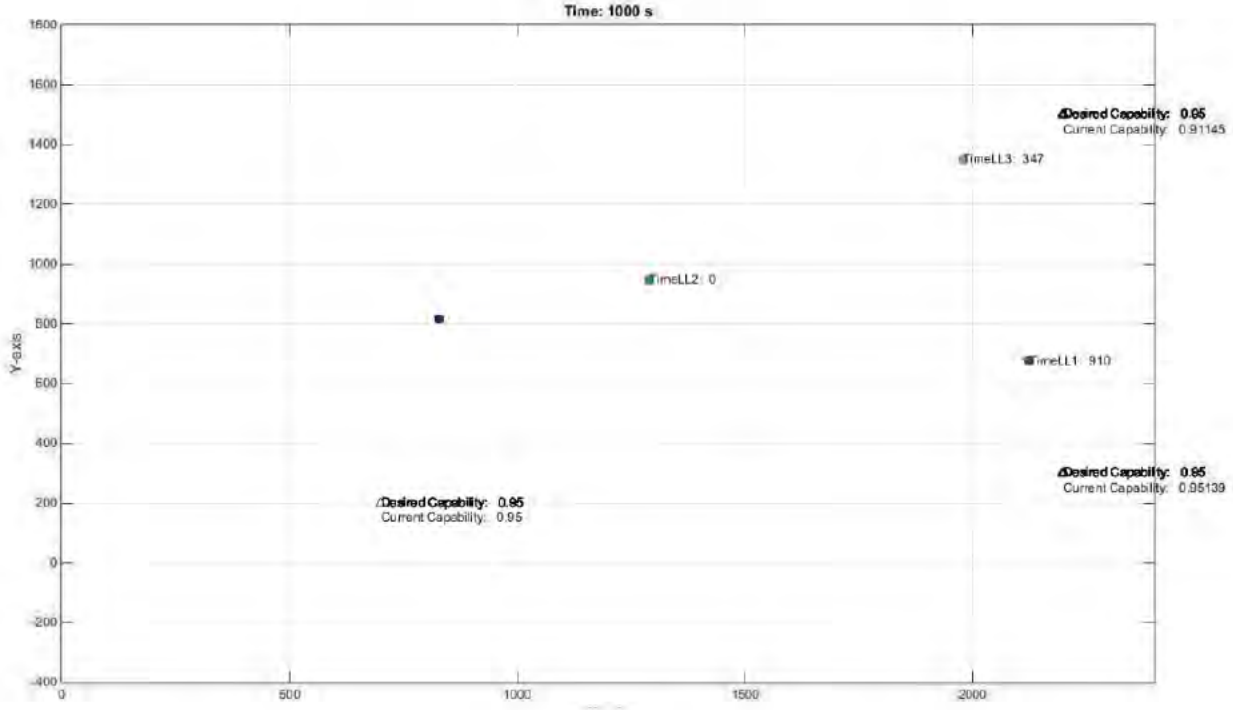

Figure 9.12: UUV 2 Gets Localized

As the simulations have shown, the UUV would move back to the USV in order to get localized and then after, it would replace the position of the UUV which has the most positional uncertainty. This cycle continues throughout the rest of the simulations, with one UUV constantly moving back to a nearby USV to get localized while the rest of the UUV provides the required capability. The only code that governs this behavior is the attractive vector incorporated into the secondary controller.

An analysis on the capability over time is as shown in Figure 9.13:

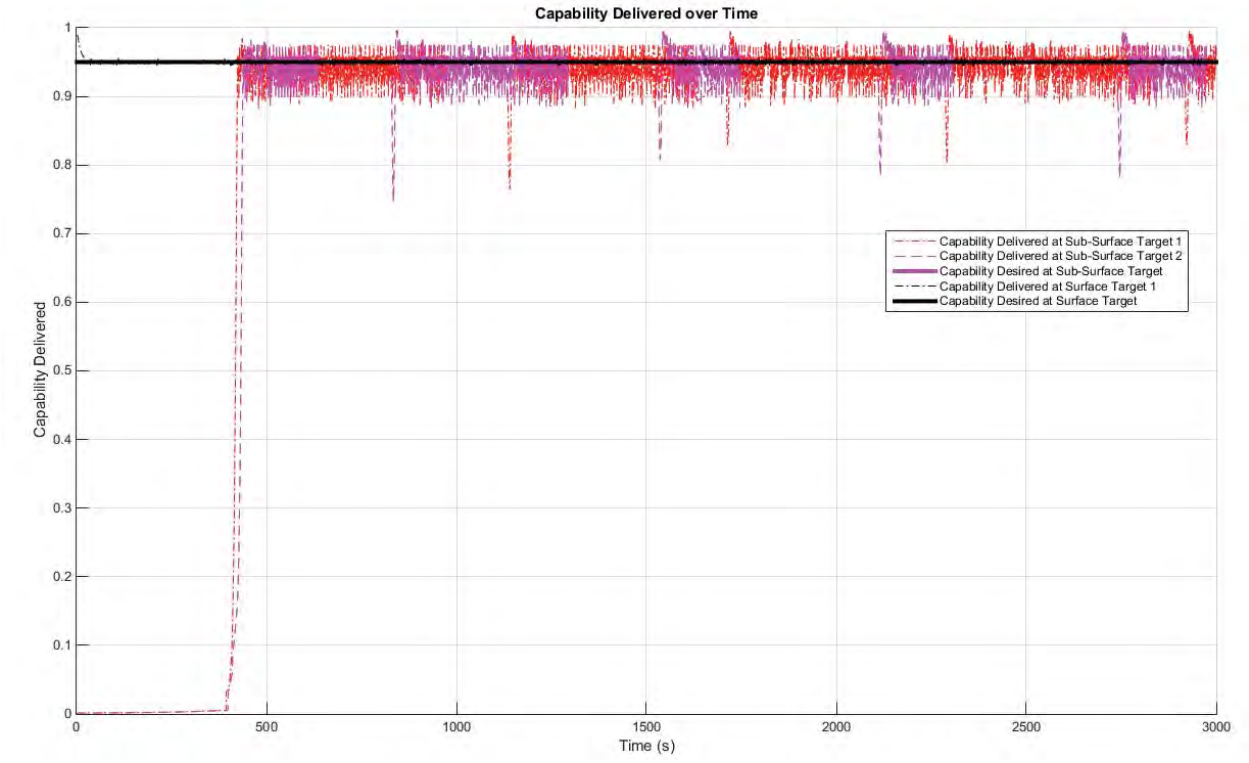

Figure 9.13: Capability Delivered over Time with CN

Overall, the incorporation of cooperative navigation into the secondary controller helps keep in check the positional uncertainty resulting in a stop to the capability degradation. As such, the control is able to provide a higher amount of capability to the system.

CHAPTER 10 – INTERDICTION

10.1 Overview

Under the current framework provided by the control system, there is capability for the swarm to detect incoming surface and sub-surface threats. However, the system lacks interdiction capability, which is definitely a requirement of vessels conducting asset protection mission. This portion of the research provides a simple interdiction system and examines how the rest of the swarm reacts with respect to the capability it provides after interdiction takes place.

10.2 Implementation of Interdiction

Ideally, preparation for interdiction should be based on a capability, which is in turn dependent on the hardware associated with the UUVs and USVs. However, given the complication associated with calculating an accurate equation that encapsulates probability of interdiction, it is assumed that interdiction is to simply have the UUV or USV intercept the target on a collision path. Interdiction is achieved for our simulations when the swarm unit reaches some nominal minimum distance from the target (which would be dictated by the exact interdiction method and hardware onboard the units).

For interdiction, the control forces the UUV closest to the target to move towards the incoming trajectory on a collision path, using an attractive vector. Interdiction takes precedence, thereby extracting the selected unit completely from the swarm's control. From the point that the unit detects the target and starts moving towards it, the interdicting UUV or USV still provides detection capability. However, once interdiction occurs, it is assumed that the UUV perishes

along with the target (or depletes its weapons system). The control system then causes the rest of the swarm to react accordingly in order to provide the required capability. In the next section, an analysis will be done on swarms of 1 USV with 3 UUVs and 1 USV with 4 UUVs, each with a demonstration of interdiction.

10.3 Interdiction with 3 UUV 1 USV Swarm

The simulation for a 3 UUV swarm was set-up as shown in Figure 10.1. This set-up is similar to the cooperative localization set-up as shown in Figure 9.5. For the purpose of the simulation, a surface threat was set to come from [4100, 1800] moving towards the swarm at a speed of x = -3 m/s.

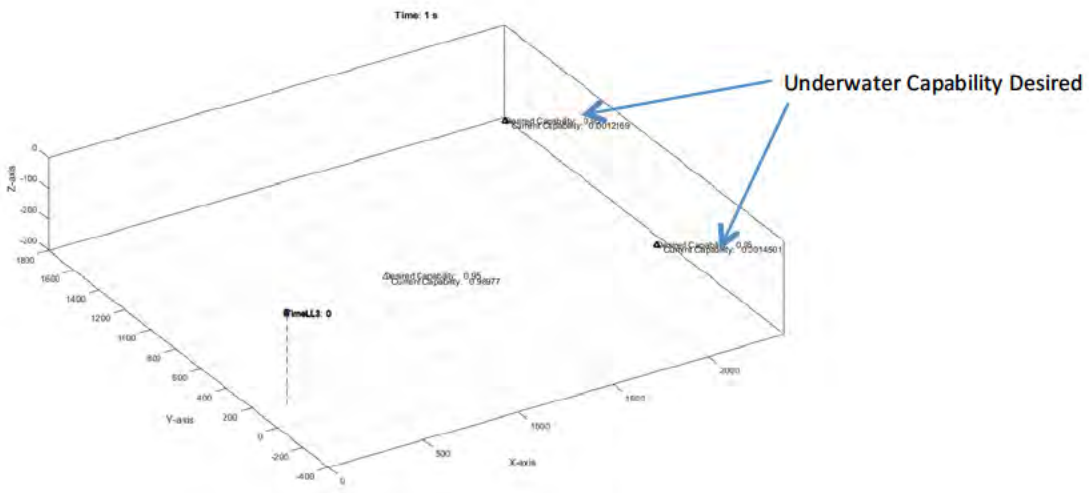

Figure 10.1: Initial Simulation Set-Up

The equations as described in chapter 3.5 were used to calculate probability of detection of the threat (the capability at the actual incoming threat location as opposed to the target location for capability). Using a random number generator to determine when detection occurs, the triggered control system has the nearest UUV move towards the threat in order to interdict. The final layout of the simulation is as shown in Figure 10.2.

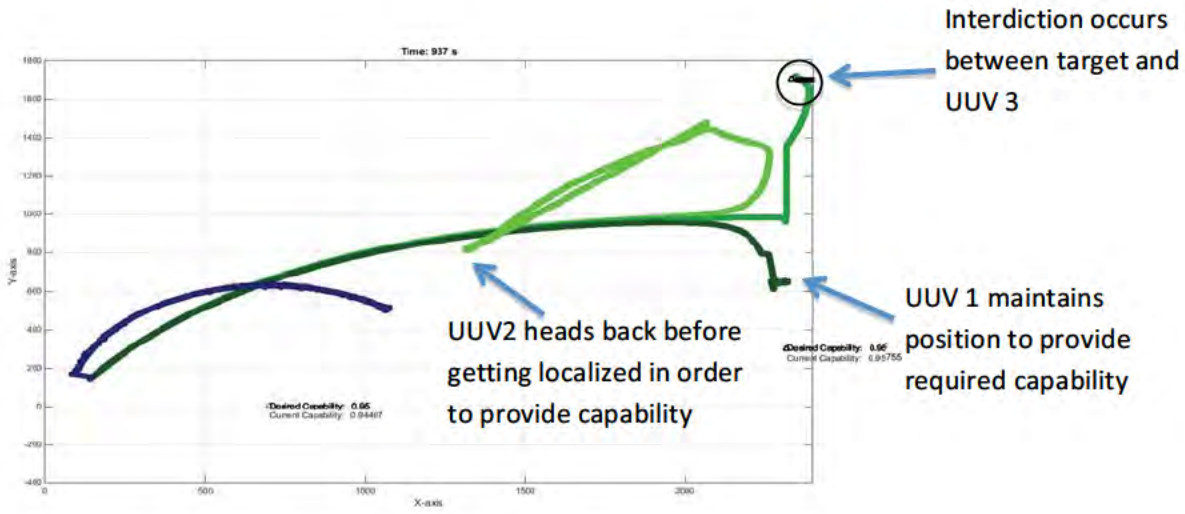

Figure 10.2: Final Simulation Set-Up

After interdiction occurs, the swarm acts as if it only has 2 UUVs left. As such, the 2 UUVs move as needed to maintain the desired capability. The particular simulation requires both UUVs to be in position to achieve the designed capability profile, and so they cannot move to be localized. Eventually, capability degradation sets in with no possibility to reposition the swarm, creating the capability over time relationship as shown in Figure 10.3.

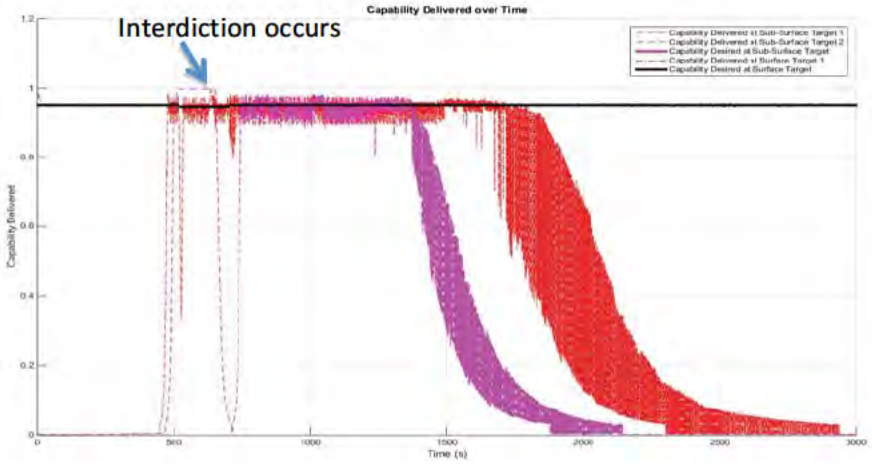

Figure 10.3: Capability over Time for Interdiction with 3 UUVs

The resulting capability degradation is less than ideal. As such, the next simulation examines interdiction in a 4 UUV swarm.

10.4 Interdiction with 4 UUV 1 USV Swarm

The following figures show a 4 UUV swarm with 1 USV to aid localization. Furthermore, it is presented in a similar top-down chronological snap shot format

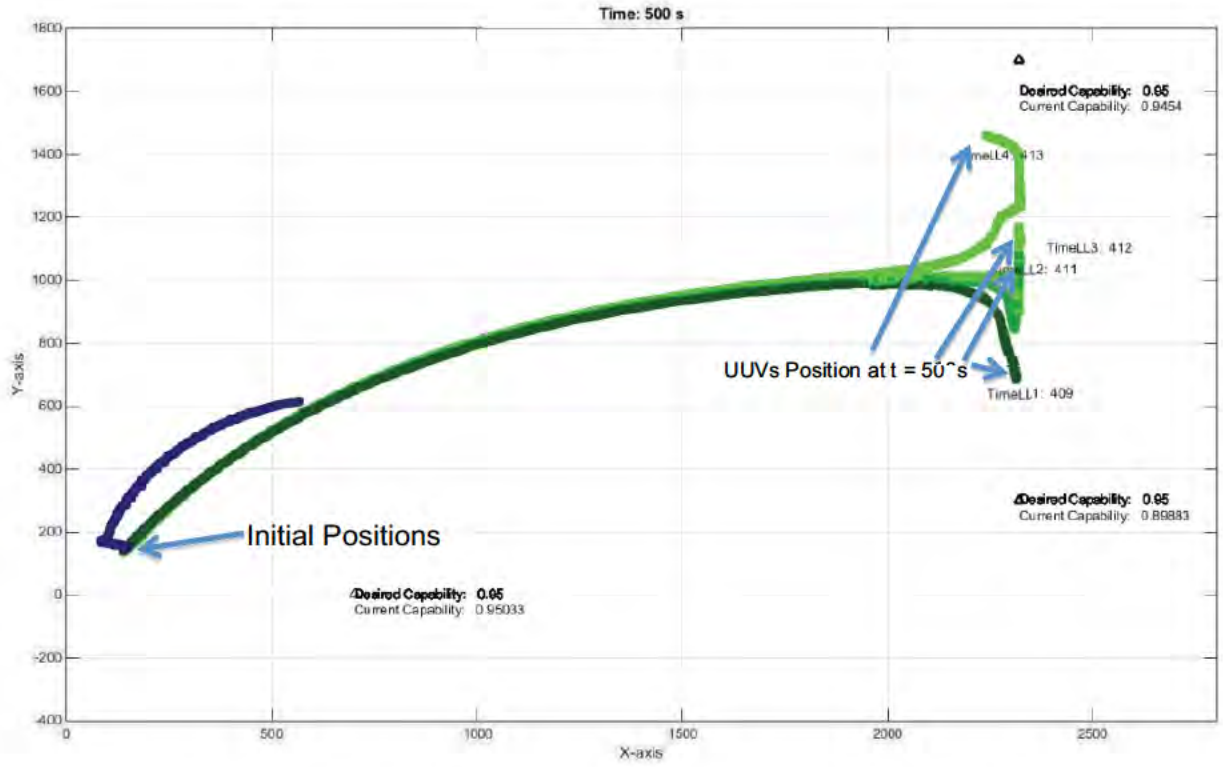

Figure 10.4: 4 UUV 1 USV Simulation with Interdiction (t = 500s)

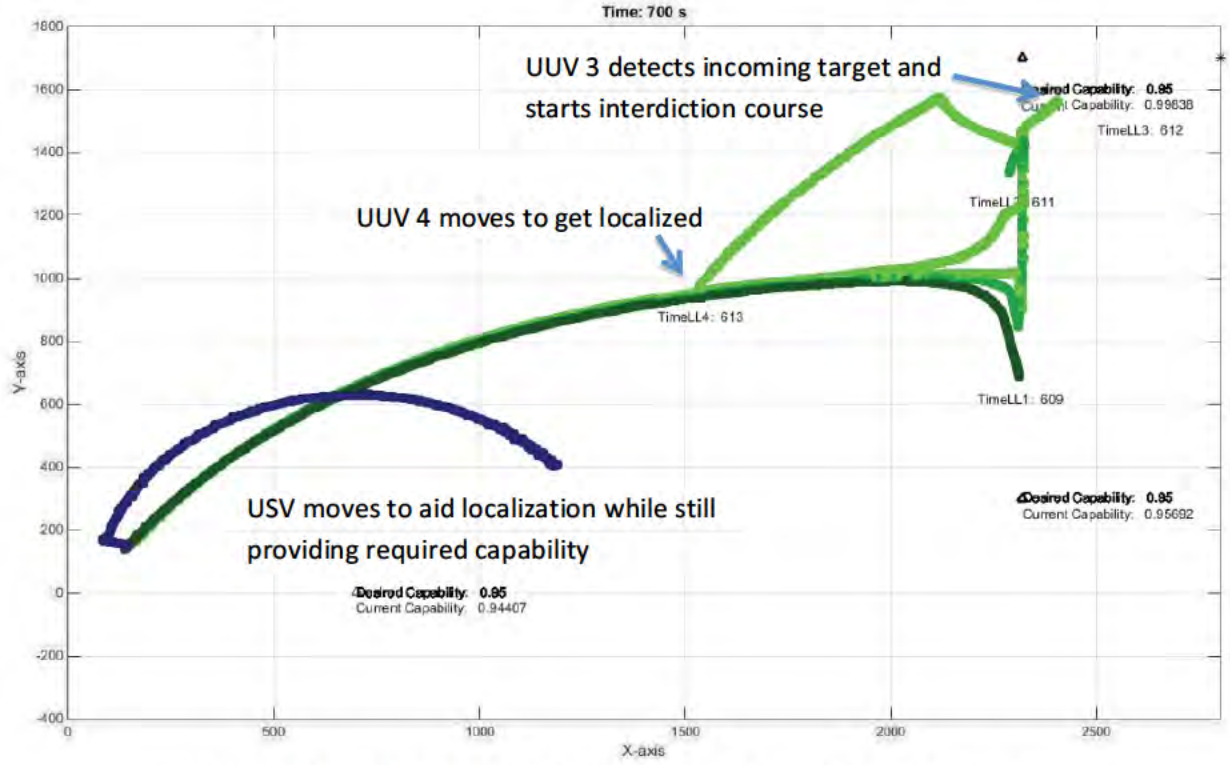

Figure 10.5: 4 UUV 1 USV Simulation with Interdiction (t = 700s)

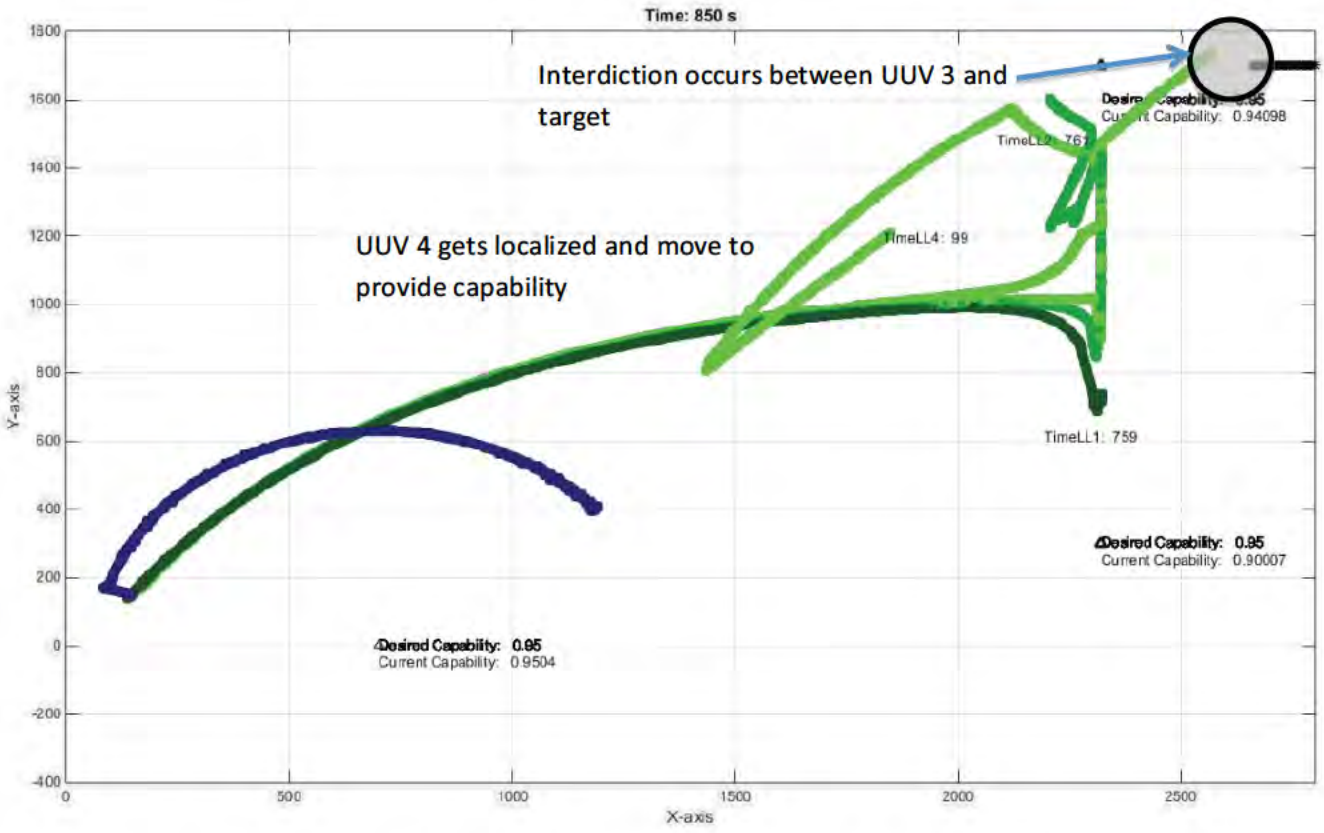

Figure 10.6: 4 UUV 1 USV Simulation with Interdiction (t = 850s)

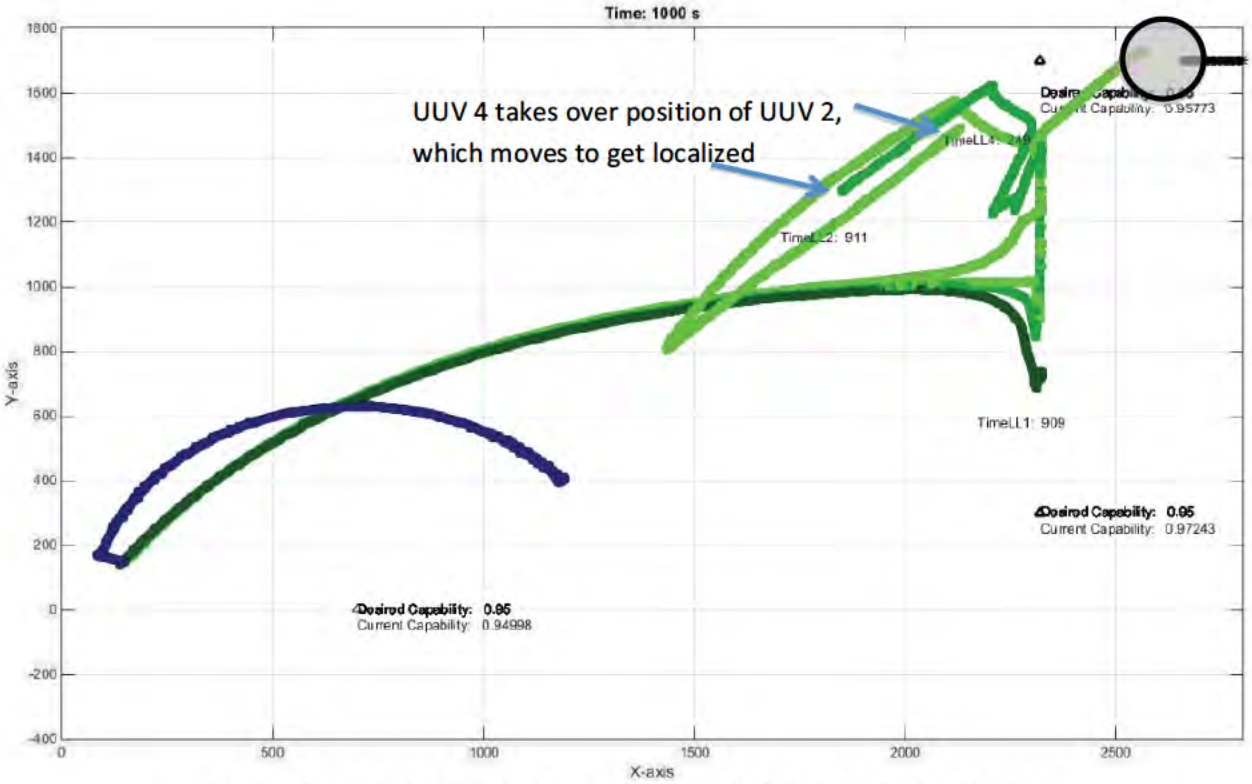

Figure 10.7: 4 UUV 1 USV Simulation with Interdiction (t = 1000s)

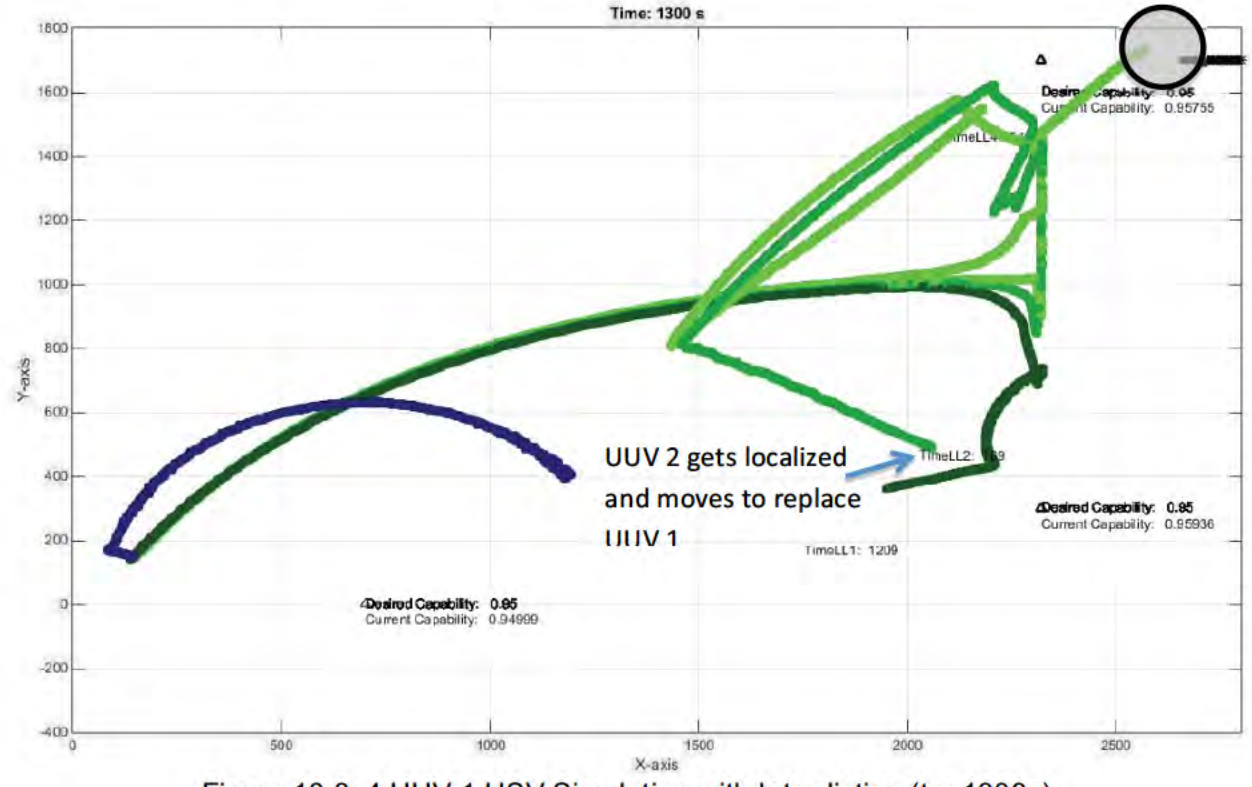

Figure 10.8: 4 UUV 1 USV Simulation with Interdiction (t = 1300s)

As seen in Figures 10.4 to 10.8, after interdiction, the swarm effectively operates as if it has 3 members and is still able to carry on with the secondary objective of cooperative navigation. As a result, the following capability plot is generated as seen in Figure 10.9.

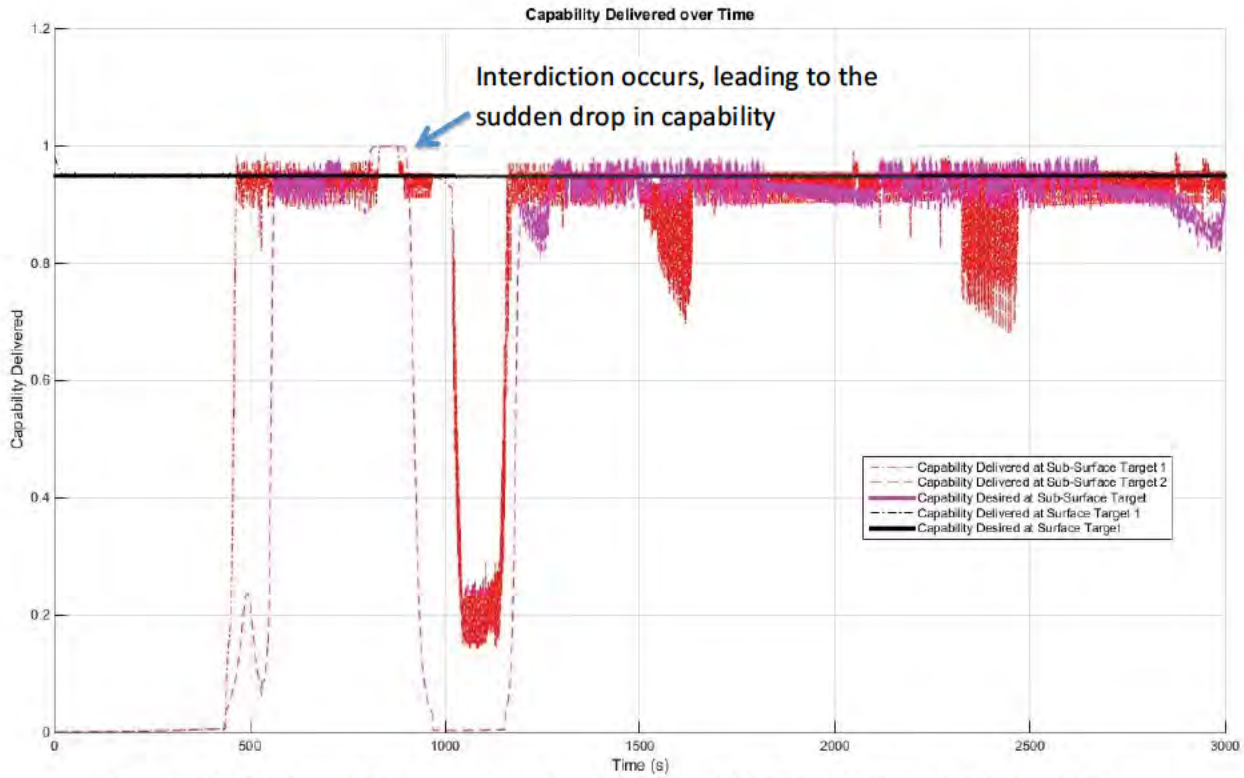

Figure 10.9: Capability over Time for 4 UUV 1 USV Simulation with Interdiction

Overall, this shows that the control system allows the incorporation of a different primary objective (in the form of interdiction), while still enabling it to cater to the primary and secondary objectives set forth in the hybrid controller. The sudden drop in capability at around 950s can be attributed to the interdiction of the target and associated reduction in the swarm overall capability as those sensors go offline.

CHAPTER 11 – CONCLUSION AND FUTURE WORKS

11.1 Contributions

In this work, a new method for cooperative control of unmanned surface vessels (USVs) and unmanned underwater vehicles (UUVs) has been demonstrated for the task of asset protection. Fundamental to this control is development of a mission-based capability concept that is grounded in realistic sensing limitations and functionality. Further, incorporation of cooperative localization was achieved using the surface vessels as aids to navigation for the underwater systems, reducing the need for surfacing to achieve localization. Finally, interdiction of targets was achieved using the same framework, with the system adapting easily to the reduction in overall capability when a unit went offline.

The primary contributions of this work are the implementation of a mission-capable hybrid control, combining systems-theoretic and behavioral control in a multi-domain framework. Furthermore, the group of UUVs and USVs were able to navigate an unknown environment successfully and position themselves such that the desired capability was met, no collisions occurred, and underwater units improved their localization performance using the surface vessels. This controller demonstrates an integrated, flexible, adaptable system capable of coordinating USV and UUV assets, and is applicable to a wide variety of missions with little modification.

11.2 Future Works

Many autonomous vehicles have been successfully deployed in recent years. However, not many of these autonomous systems work cooperatively in different domains and attempt to synergize with each other. The effectiveness and adaptability shown in this form of controller does indeed seem truly promising and, in the long run, may prove to be a major keystone for swarm control.

In the short run, future work could build on this form of control in order to further enhance the cooperative element between USVs and UUVs. Furthermore, practical trials utilizing the necessary hardware and based on this form of control could indeed prove fruitful in driving research in the field of swarm control as well.

There is no doubt that this research is promising and has tremendous potential to impact the world of robotics. With the necessary control system, a robust, effective and adaptable swarm will be able to fulfill a set of secondary missions while ensuring that the primary pre-requisites are met as well.

APPENDIX A –CODE FOR SIMPLE SURFACE PRIMARY SIMULATION

```
% Input Variables
cap desired   0.95;
xd   2000;
yd   2000;

% Global variables
res req   3;                % Required pixels for detection of Small Boat via EO
height   4.5;
horizon   (sqrt(17*height)+sqrt(17*height)) * 1000;

% Create USVs
x1   1;
y1   1;
x2   50;
y2   50;
swarmstate   [x1;y1;x2;y2];
x1 plot(1)   1
y1 plot(1)   1
x2 plot(1)   50
y2 plot(1)   50


for t   1:1:50

    % Capability Calculations
    if ((640/(2*sqrt((yd   y1)^2 + (xd   x1)^2)*tand(24/2))) > 3) & ((640/(2*sqrt((yd   y2)^2 +
(xd   x2)^2)*tand(24/2))) > 3)

        c1   1   0.00000000000000001*sqrt((yd   y1)^2 + (xd   x1)^2);
        c2   (1   0.00000000000000001*sqrt((yd   y2)^2 + (xd   x2)^2));
        syms y1s y2s x1s x2s;

        jacobian s   jacobian((1   0.00000000000000001*sqrt((yd   y1s)^2 + (xd   x1s)^2))*(1
0.00000000000000001*sqrt((yd   y2s)^2 + (xd   x2s)^2)) +(1   0.00000000000000001*sqrt((yd   y1s)^2
+ (xd   x1s)^2))*(1 (1   0.00000000000000001*sqrt((yd   y2s)^2 + (xd   x2s)^2))) + (1
0.00000000000000001*sqrt((yd   y2s)^2 + (xd   x2s)^2))*(1 (1   0.00000000000000001*sqrt((yd
y1s)^2 + (xd   x1s)^2))),[x1s,y1s,x2s,y2s]);
        jacobian subbed   subs(jacobian s,[x1s,y1s,x2s,y2s],[x1,y1,x2,y2]);
        jacobian   double(jacobian subbed);       % CONVERT SYMBOLS BACK TO NUMBERS
    elseif (640/(2*sqrt((yd   y1)^2 + (xd   x1)^2)*tand(24/2))) > 3
        c1   (1   0.00000000000000001*sqrt((yd   y1)^2 + (xd   x1)^2));
        c2   sin((pi/2)*(640/(2*sqrt((yd   y2)^2 + (xd   x2)^2)*tand(24/2)))/(3));
        syms y1s y2s x1s x2s;

        jacobian s   jacobian((1   0.00000000000000001*sqrt((yd   y1s)^2 + (xd
x1s)^2))*sin((pi/2)*(640/(2*sqrt((yd   y2s)^2 + (xd   x2s)^2)*tand(24/2)))/(3))+(1
sin((pi/2)*(640/(2*sqrt((yd   y2s)^2 + (xd   x2s)^2)*tand(24/2)))/(3)))*(1
0.00000000000000001*sqrt((yd   y1s)^2 + (xd   x1s)^2))+(1 (1   0.00000000000000001*sqrt((yd
y1s)^2 + (xd   x1s)^2)))*sin((pi/2)*(640/(2*sqrt((yd   y2s)^2 + (xd
x2s)^2)*tand(24/2)))/(3)),[x1s,y1s,x2s,y2s]);
        jacobian subbed   subs(jacobian s,[x1s,y1s,x2s,y2s],[x1,y1,x2,y2]);
        jacobian   double(jacobian subbed);


    elseif (640/(2*sqrt((yd   y2)^2 + (xd   x2)^2)*tand(24/2))) > 3
        c2   (1   0.00000000000000001*sqrt((yd   y2)^2 + (xd   x2)^2));
        c1   sin((pi/2)*(640/(2*sqrt((yd   y1)^2 + (xd   x1)^2)*tand(24/2)))/(3));
        syms y1s y2s x1s x2s;

        jacobian s   jacobian(sin((pi/2)*(640/(2*sqrt((yd   y1s)^2 + (xd
x1s)^2)*tand(24/2)))/(3))*(1   0.00000000000000001*sqrt((yd   y2s)^2 + (xd   x2s)^2)) + (1 (1
0.00000000000000001*sqrt((yd   y2s)^2 + (xd   x2s)^2)))*sin((pi/2)*(640/(2*sqrt((yd   y1s)^2 + (xd
  x1s)^2)*tand(24/2)))/(3)) + (1 sin((pi/2)*(640/(2*sqrt((yd   y1s)^2 + (xd
```

```
x1s)^2)*tand(24/2)))/(3)))*((1   0.0000000000000001*sqrt((yd   y2s)^2 + (xd
x2s)^2))),[x1s,y1s,x2s,y2s]);
        jacobian subbed   subs(jacobian s,[x1s,y1s,x2s,y2s],[x1,y1,x2,y2]);
        jacobian   double(jacobian subbed);

    else
        c1   (sin((pi/2)*(640/(2*sqrt((yd   y1)^2 + (xd   x1)^2)*tand(24/2)))/(3)));
        c2   (sin((pi/2)*(640/(2*sqrt((yd   y2)^2 + (xd   x2)^2)*tand(24/2)))/(3)));
        syms y1s y2s x1s x2s;

        jacobian s   jacobian(((sin((pi/2)*(640/(2*sqrt((yd   y1s)^2 + (xd
x1s)^2)*tand(24/2)))/(3)))*(sin((pi/2)*(640/(2*sqrt((yd   y2)^2 + (xd
x2s)^2)*tand(24/2)))/(3)))) + ((1 (sin((pi/2)*(640/(2*sqrt((yd   y2s)^2 + (xd
x2s)^2)*tand(24/2)))/(3))))*(sin((pi/2)*(640/(2*sqrt((yd   y1s)^2 + (xd
x1s)^2)*tand(24/2)))/(3)))) + ((1 (sin((pi/2)*(640/(2*sqrt((yd   y1s)^2 + (xd
x1s)^2)*tand(24/2)))/(3))))*(sin((pi/2)*(640/(2*sqrt((yd   y2s)^2 + (xd
x2s)^2)*tand(24/2)))/(3)))),[x1s,y1s,x2s,y2s]);
        jacobian subbed   subs(jacobian s,[x1s,y1s,x2s,y2s],[x1,y1,x2,y2]);
        jacobian   double(jacobian subbed);
    end

    % Overall Capability
    overall cap   c1*c2 + c1*(1 c2) + c2*(1 c1);

    % Control Code
    jacob pinv   pinv(jacobian);
    cdot   0.095*(cap desired   overall cap);
    swarmstatedot   jacob pinv*cdot;
    swarmstate   swarmstate + swarmstatedot;

    x1   swarmstate(1);
    x2   swarmstate(3);
    y1   swarmstate(2);
    y2   swarmstate(4);

    x1 plot(t+1)   swarmstate(1);
    y1 plot(t+1)   swarmstate(2);
    x2 plot(t+1)   swarmstate(3);
    y2 plot(t+1)   swarmstate(4);
    cap(t)   overall cap;
    line(t)   cap desired;

end

% Plots
figure(1)
hold on
grid on
plot(x1 plot,y1 plot,'*')
plot(x2 plot,y2 plot,'r+')
plot(xd,yd,'^k')
xlabel('X Axis (m)')
ylabel('Y Axis (m)')
title('Simulation of Primary Control')

figure(2)
hold on
plot(0:49,cap)
plot(0:49,line)
xlabel('Time (s)')
ylabel('Capability Delivered')
title('Capability Delivered Over Time')
legend('Capability Delieverd','Capability Desired')
```

APPENDIX B –CODE FOR SURFACE HYBRID CONTROL

```
% Input Variables
cap desired   [0.95;0.95];
xd   [4000;4000];
yd   [1000;2000];

% Global variables
res req   3;                % Required pixels for detection of Small Boat via EO
height   4.5;
horizon   (sqrt(17*height)+sqrt(17*height)) * 1000;
vunit   [0;0;0;0;0;0;0;0]

% Obstacles

x1 obs   2.5954e+03;
y1 obs     1.4982e+03;

x2 obs   1.7503e+03;
y2 obs   1.3404e+03;

x3 obs    702.8923;
y3 obs    788.8338;

obs mat   [x1 obs;y1 obs;x2 obs;y2 obs;x3 obs;y3 obs]

% Create USVs
x1   150.2;
y1   150.2;
x2   150.1;
y2   150.1;
x3   150;
y3   150;
x4   149.999;
y4   149.999;

swarmstate   [x1;y1;x2;y2;x3;y3;x4;y4];
x1 plot(1)   x1;
y1 plot(1)   y1;
x2 plot(1)   x2;
y2 plot(1)   y2;
x3 plot(1)   x3;
y3 plot(1)   y3;
x4 plot(1)   x4;
y4 plot(1)   y4;
syms c symbol x1s y1s x2s y2s x3s y3s x4s y4s
ss   [x1s;y1s;x2s;y2s;x3s;y3s;x4s;y4s];

for t   1:1:700  % t   time in seconds
    for i   1:1:length(cap desired)           % i   amount of capabilities desired

        vobst   [0;0;0;0;0;0;0;0];


        % CALCULATE CAPABILITIES FOR EACH INDIVIDUAL VESSELS FOR SPECIFIC POINT
        for n   1:1:((length(swarmstate))/2)    % n   vessel number
            if sqrt(((yd(i)   swarmstate(n*2))^2) + ((xd(i)   swarmstate(((n*2) 1)))^2)) >
horizon
                c(n)   100/((sqrt(((yd(i)   swarmstate(n*2))^2) + ((xd(i)   swarmstate(((n*2)
1)))^2))));

                c symbol(n)   100/((sqrt(((yd(i)   ss(n*2))^2) + ((xd(i)   ss(((n*2) 1))))^2)));

            elseif ((640/(2*sqrt((yd(i)   swarmstate(n*2))^2 + (xd(i)   swarmstate(((n*2)
1)))^2)*tand(24/2))) > 3)
```

```
                c(n)    1   0.0001*sqrt((yd(i)   swarmstate(n*2))^2 + (xd(i)   swarmstate(((n*2)
1)))^2);

                c symbol(n)    1   0.0001*sqrt((yd(i)   ss(n*2))^2 + (xd(i)   ss(((n*2) 1)))^2);
            else
                c(n)   sin((pi/2)*(640/(2*sqrt((yd(i)   swarmstate(n*2))^2 + (xd(i)
swarmstate(((n*2) 1)))^2)*tand(24/2)))/(3));

                c symbol(n)   sin((pi/2)*(640/(2*sqrt((yd(i)   ss(n*2))^2 + (xd(i)   ss(((n*2)
1)))^2)*tand(24/2)))/(3));
            end
        end



        % CALCULATE OVERALL CAPABILITY
        capability(i)   0 ;
        cadd   1;
        for o   1:1:((length(swarmstate))/2)
            capability(i)   capability(i) + c(o)*cadd;
            cadd   (1   capability(i));
        end

        overall cap   0 ;
        cadd   1;
        for o   1:1:((length(swarmstate))/2)
            overall cap   overall cap + c symbol(o)*cadd;
            cadd   (1   overall cap);
        end



        % SUB CAPABILITY(SYMBOLS) INTO JACOBIAN
        jacobian s   jacobian(overall cap,[x1s,y1s,x2s,y2s,x3s,y3s,x4s,y4s]);       % calculate
jacobian matrix based on partial derivative of position 0f (x1, y1)...
        jacobian subbed
subs(jacobian s,[x1s,y1s,x2s,y2s,x3s,y3s,x4s,y4s],[swarmstate(1),swarmstate(2),swarmstate(3),swar
mstate(4),swarmstate(5),swarmstate(6),swarmstate(7),swarmstate(8)]);
        jaco mat(i,:)   (double(jacobian subbed)); % final jacobian in numbers

    end

    % ADD IN OBSTACLE AVOIDANCE

    safezone   100;

    for n   1:1:((length(swarmstate))/2)    % n   vessel number
        objrepx   0;
        objrepy   0;
        for obs   1:1:((length(obs mat)/2))   % obs   number of obstructions
            dist   sqrt((swarmstate((n*2) 1)   obs mat((obs*2) 1))^2 + (swarmstate((n*2))
obs mat((obs*2)))^2);

            if dist < safezone
                angle   atan2((swarmstate((n*2))   obs mat((obs*2))),(swarmstate((n*2) 1)
obs mat((obs*2) 1)));
                objrepx   objrepx + 4000*((1/dist) (1/safezone))*cos(angle);
                objrepy   objrepy + 4000*((1/dist) (1/safezone))*sin(angle);
            end
            vobst rec(:,t)   vobst;

        end
        vobst((n*2) 1)  objrepx;
        vobst(n*2)  objrepy;

    end

    %      ADD IN UNIT AVOIDANCE
```

```
    safezone unit   100;

    for n   1:1:((length(swarmstate))/2)    % n   vessel number
        unitrepx   0;
        unitrepy   0;
        for o   1:1:((length(swarmstate))/2)    % o   number of vessels

            if n    o
                vunit((n*2) 1)  0;
                vunit(n*2)  0;
            else
                dist unit   sqrt((swarmstate((n*2) 1)   swarmstate((o*2) 1))^2 +
(swarmstate((n*2))   swarmstate((o*2)))^2);

                if dist unit < safezone unit
                    angle unit   atan2((swarmstate((n*2))   swarmstate(o*2)),(swarmstate((n*2) 1)
 swarmstate((o*2) 1)));
                    unitrepx   unitrepx + 4000*((1/dist unit) (1/safezone unit))*cos(angle unit);
                    unitrepy   unitrepy + 4000*((1/dist unit) (1/safezone unit))*sin(angle unit);
                end
            end


        end
        vunit((n*2) 1)  unitrepx;
        vunit(n*2)  unitrepy;

    end

    % CONTROL LAW
    jacob pinv   pinv(jaco mat);
    capability t   transpose(capability);
    capability plot(t,:)   capability;
    cdot   0.009*(cap desired   capability t)
    [sizex, sizey]   size(jacob pinv*jaco mat);
    swarmstatedot   jacob pinv*cdot + 150*((eye(sizex,sizey)   jacob pinv*jaco mat)*vobst) +
     150*((eye(sizex,sizey)   jacob pinv*jaco mat)*vunit);

    % LIMITATIONS FOR SPEED
    for n   1:1:((length(swarmstate))/2)
        while abs(sqrt(((swarmstatedot((n*2) 1))^2) + (swarmstatedot(n*2)^2))) > 50
            if swarmstatedot((n*2) 1) >  0
                swarmstatedot((n*2) 1)   0.1*swarmstatedot((n*2) 1);
            else
                swarmstatedot((n*2) 1)   0.1*swarmstatedot((n*2) 1);
            end

            if swarmstatedot(n*2) >  0
                swarmstatedot(n*2)   0.1*swarmstatedot(n*2);
            else
                swarmstatedot(n*2)   0.1*swarmstatedot(n*2);
            end
        end
    end

    % CONTROL LAW (CONT'D)
    swarmstate   swarmstate + swarmstatedot;

    x1 plot(t+1)   swarmstate(1);
    y1 plot(t+1)   swarmstate(2);
    x2 plot(t+1)   swarmstate(3);
    y2 plot(t+1)   swarmstate(4);
    x3 plot(t+1)   swarmstate(5);
    y3 plot(t+1)   swarmstate(6);
    x4 plot(t+1)   swarmstate(7);
    y4 plot(t+1)   swarmstate(8);

    cap 1(t)   capability(1)
    cap 2(t)   capability(2)
```

```
end

% PLOT
figure(1)
hold on
axis equal
grid on
plot(x1 plot,y1 plot,'b*')
plot(x2 plot,y2 plot,'y*')
plot(x3 plot,y3 plot,'g*')
plot(x4 plot,y4 plot,'k*')
plot(x1 obs,y1 obs,'rd','LineWidth',6.5)
plot(x2 obs,y2 obs,'rd','LineWidth',6.5)
plot(x3 obs,y3 obs,'rd','LineWidth',6.5)

plot(xd,yd,'^k')
xlabel('X Axis')
ylabel('Y Axis')
title('Simulation Run for t   700')

figure(2)
hold on
grid on
plot(1:length(cap 1),cap 1);
plot(1:length(cap 2),cap 2);
plot([1 length(cap 2)],[0.95 0.95]);
```

APPENDIX C – CODE FOR SUB-SURFACE HYBRID CONTROL

```
% Input Variables

cap desired U   [0.95; 0.95];
xd U   [4000; 800];
yd U   [4000;2000];
zd U   [ 400;  900];

% Global Variables

pfa   0.01;  % Assuming Probablity of False Alarm as 0.01
sub 1   erfcinv(2*pfa);
f   150;                                % Frequency of Sonar
alpha   (0.036*f^2)/(f^2 + 3600)+(3.2 * 10^ 7 * f^2);

% Obstacles

x1 obs u   1.8439e+03;
y1 obs u   1.0135e+03;
z1 obs u      21.6312;

x2 obs u     2.1409e+03;
y2 obs u    2.1409e+03;
z2 obs u     288.5518;


obs mat u   [x1 obs u;y1 obs u;z1 obs u;x2 obs u;y2 obs u;z2 obs u;]
vobst u   [0;0;0;0;0;0;0;0;0]
vunit u   [0;0;0;0;0;0;0;0;0]



% Create USVs
x1 U   149;
y1 U   149;
z1 U    1;
x2 U   148.5;
y2 U   148.5;
z2 U    1.4;
x3 U   149.5;
y3 U   149.5;
z3 U    1.8;
swarmstate U   [x1 U;y1 U;z1 U;x2 U;y2 U;z2 U; x3 U;y3 U;z3 U];
syms snr s magnitude s TL s x1u y1u z1u x2u y2u z2u x3u y3u z3u
ssu   [x1u;y1u;z1u;x2u;y2u;z2u;x3u;y3u;z3u];
x1 plot u(1)   x1 U;
y1 plot u(1)   y1 U;
z1 plot u(1)   z1 U;
x2 plot u(1)   x2 U;
y2 plot u(1)   y2 U;
z2 plot u(1)   z2 U;
x3 plot u(1)   x3 U;
y3 plot u(1)   y3 U;
z3 plot u(1)   z3 U;

for t   1:1:450
    for i   1:1:length(cap desired U)
        for n   1:1:((length(swarmstate U))/3)    % n   vessel number

            R   sqrt((xd U(i)   (swarmstate U(n*3   2)))^2 + (yd U(i)   swarmstate U(n*3   1))^2
+ (zd U(i)   swarmstate U(n*3))^2);
            R s   sqrt((xd U(i)   (ssu(n*3   2)))^2 + (yd U(i)   ssu(n*3   1))^2 + (zd U(i)
ssu(n*3))^2);
            if R > 501
                pd(n)   1/R;
```

```
                pd symbol(n)   1/R s;
            else

                TL   10*log(R) + 30 + alpha*R;        % Transmission Loss based on Cylindrical
Spreading
                TS   10*log(0.25*10^2);               % Target Strength based on large sphere w/
10m radius
                SL   180;                             % Assuming Sonar Source Level of 180db
                snr   SL   2*TL + TS;
                magnitude   10^(snr/10);
                fhandle   @(x) exp( x.^2);
                pd(n)   0.5 * (2/sqrt(pi))*integral(fhandle,sub 1 sqrt(magnitude),Inf);


                TL s    10*log(R s) + 30 + alpha*R s;
                snr s   SL   2*TL s + TS;
                magnitude s   10^(snr s/10);
                pd symbol(n)   0.5 * (2/sqrt(pi))*int(fhandle,sub 1 sqrt(magnitude s),Inf);
            end

        end
        % calculate overall capability
        capability u(i)   0;
        cadd u   1;
        for o   1:1:((length(swarmstate U))/3)
            capability u(i)   capability u(i) + pd(o)*cadd u;
            cadd u   (1   capability u(i));
        end

        overall cap u 0;
        cadd   1;
        for o   1:1:((length(swarmstate U))/3)
            overall cap u   overall cap u + pd symbol(o)*cadd;
            cadd   (1   overall cap u);
        end

        jacobian s u   jacobian(overall cap u,[x1u,y1u,z1u,x2u,y2u,z2u,x3u,y3u,z3u]);       %
calculate jacobian matrix based on partial derivative of position 0f (x1, y1)...
        jacobian subbed u
subs(jacobian s u,[x1u,y1u,z1u,x2u,y2u,z2u,x3u,y3u,z3u],[swarmstate U(1),swarmstate U(2),swarmsta
te U(3),swarmstate U(4),swarmstate U(5),swarmstate U(6),swarmstate U(7),swarmstate U(8),swarmstat
e U(9)]);
        jaco mat u(i,:)   (double(jacobian subbed u)); % final jacobian in numbers

    end

    % ADD IN OBSTACLE AVOIDANCE

    safezone u   100;

    for n   1:1:((length(swarmstate U))/3)    % n   vessel number
        objrepx u   0;
        objrepy u   0;
        objrepz u   0;
        for u   1:1:((length(obs mat u)/3))   % obs   number of obstructions
            dist u   sqrt((swarmstate U(n*3   2) obs mat u(u*3 2))^2 + (swarmstate U(n*3   1)
obs mat u(u*3  1))^2 + (swarmstate U(n*3) obs mat u(u*3))^2);

            if dist u < safezone u
                angle u alpha   atan2((swarmstate U((n*3 2))   obs mat u((u*3
2))),(swarmstate U((n*3) 1)   obs mat u(u*3 1)));
                angle u beta   atan2((swarmstate U((n*3))
obs mat u((u*3))),(swarmstate U((n*3) 1)   obs mat u(u*3 1)));
                angle u charlie   atan2((swarmstate U((n*3))
obs mat u((u*3))),(swarmstate U((n*3) 2)   obs mat u(u*3 2)));

                objrepx u   objrepx u + 300*((1/dist u) (1/safezone u))*cos(angle u alpha) +
300*((1/dist u) (1/safezone u))*cos(angle u charlie);
```

```
                objrepy u   objrepy u + 300*((1/dist u) (1/safezone u))*sin(angle u alpha) +
300*((1/dist u) (1/safezone u))*cos(angle u beta);
                objrepz u   objrepz u + 300*((1/dist u) (1/safezone u))*sin(angle u beta) +
300*((1/dist u) (1/safezone u))*sin(angle u charlie);
            end

        end
        vobst u((n*3) 2)  objrepx u;
        vobst u(n*3  1)  objrepy u;
        vobst u(n*3)  objrepz u;


    end

    % ADD IN UNIT AVOIDANCE

    safezone unit u   100;

    for n   1:1:((length(swarmstate U))/3)    % n   vessel number
        unitrepx u   0;
        unitrepy u   0;
        unitrepz u   0;
        for o   1:1:((length(swarmstate U))/3)    % o   number of vessels

            if n    o
                vunit u((n*3) 2)  0;
                vunit u((n*3) 1)  0;
                vunit u(n*3)  0;
            else
                dist unit u   sqrt((swarmstate U(n*3   2) swarmstate U(o*3   2))^2 +
(swarmstate U(n*3   1) swarmstate U(o*3   1))^2 + (swarmstate U(n*3) swarmstate U(o*3))^2);

                if dist unit u < safezone unit u

                    angle unit u alpha   atan2((swarmstate U((n*3 2))   swarmstate U((o*3
2))),(swarmstate U((n*3) 1)   swarmstate U(o*3 1)));
                    angle unit u beta   atan2((swarmstate U((n*3))
swarmstate U((o*3))),(swarmstate U((n*3) 1)   swarmstate U(o*3 1)));
                    angle unit u charlie   atan2((swarmstate U((n*3))
swarmstate U((o*3))),(swarmstate U((n*3) 2)   swarmstate U(o*3 2)));



                    unitrepx u   unitrepx u + 800*((1/dist unit u)
(1/safezone unit u))*cos(angle unit u alpha)+ 800*((1/dist unit u)
(1/safezone unit u))*cos(angle unit u charlie);
                    unitrepy u   unitrepy u + 800*((1/dist unit u)
(1/safezone unit u))*sin(angle unit u alpha)+ 800*((1/dist unit u)
(1/safezone unit u))*cos(angle unit u beta);
                    unitrepz u   unitrepz u + 800*((1/dist unit u)
(1/safezone unit u))*sin(angle unit u beta)+ 800*((1/dist unit u)
(1/safezone unit u))*sin(angle unit u charlie);

                end

            end
            vunit u((n*3) 2)  unitrepx u;
            vunit u((n*3)  1)  unitrepy u;
            vunit u(n*3)  unitrepz u;


        end
    end

    % CONTROL LAW

    jacob pinv u   pinv(jaco mat u);
    capability u t  transpose(capability u);
    capability plot u(t,:)   capability u;
```

```
    cdot u   0.09*(cap desired U   capability u t);
    [sizex u, sizey u]   size(jacob pinv u*jaco mat u);

    swarmstatedot u   jacob pinv u * cdot u + 45000*((eye(sizex u,sizey u)
jacob pinv u*jaco mat u)*vobst u) + 45000*((eye(sizex u,sizey u)
jacob pinv u*jaco mat u)*vunit u);

    % Speed Limitations

    for n   1:1:((length(swarmstate U))/3)
        while abs(sqrt(((swarmstatedot u((n*3) 2))^2) + (swarmstatedot u((n*3) 1)^2) +
swarmstatedot u(n*3)^2)) > 30
            if swarmstatedot u((n*3) 2) >  0
                swarmstatedot u((n*3) 2)   0.5*swarmstatedot u((n*3) 2);            else
                swarmstatedot u((n*3) 2)   0.5*swarmstatedot u((n*3) 2);
            end

            if swarmstatedot u((n*3) 1) >  0
                swarmstatedot u((n*3) 1)   0.5*swarmstatedot u((n*3) 1);
            else
                swarmstatedot u((n*3) 1)   0.5*swarmstatedot u((n*3) 1);
            end

            if swarmstatedot u(n*3) >  0
                swarmstatedot u(n*3)   0.5*swarmstatedot u(n*3);
            else
                swarmstatedot u(n*3)   0.5*swarmstatedot u(n*3);
            end
        end
    end

    swarmstate U   swarmstate U + swarmstatedot u;
    for n   1:1:((length(swarmstate U))/3)    % n   vessel number
        if swarmstate U(n*3) > 0
            swarmstate U(n*3)   0;
        end
    end
    x1 plot u(t+1)   swarmstate U(1);
    y1 plot u(t+1)   swarmstate U(2);
    z1 plot u(t+1)   swarmstate U(3);
    x2 plot u(t+1)   swarmstate U(4);
    y2 plot u(t+1)   swarmstate U(5);
    z2 plot u(t+1)   swarmstate U(6);
    x3 plot u(t+1)   swarmstate U(7);
    y3 plot u(t+1)   swarmstate U(7);
    z3 plot u(t+1)   swarmstate U(9);

    capability(t)   capability u(1);
    capability 2(t)   capability u(2);
end

figure(1)
hold on
grid on
plot3(transpose(x1 plot u),transpose(y1 plot u),transpose(z1 plot u),'c*')
plot3(transpose(x2 plot u),transpose(y2 plot u),transpose(z2 plot u),'gd')
plot3(transpose(x3 plot u),transpose(y3 plot u),transpose(z3 plot u),'ko')

plot3(xd U,yd U,zd U,'k^','LineWidth',2.0)
plot3(x1 obs u,y1 obs u,z1 obs u,'rd','LineWidth',6.0)
plot3(x2 obs u,y2 obs u,z2 obs u,'rd','LineWidth',6.0)

xlabel('X axis')
ylabel('Y axis')
zlabel('Z axis')
title('3D Simulations with Primary Control for 3 Unit Swarm')
legend('Path of USV 1','Path of USV 2', 'Path of USV 3', 'Target Positions','Obstacles')

figure(2)
```

```
hold on
grid on
plot(1:length(capability),capability)
plot(1:length(capability 2),capability 2)
plot([1 length(capability)],[0.95 0.95])
xlabel('Time (s)')
ylabel('Capability Delivered')
title('Capability Delivered over Time')
legend('Capability Delivered at Target 1','Capability Delivered at Target 2', 'Capability
Desired')
```

APPENDIX D – CODE FOR SURFACE AND SUB-SURFACE HYBRID CONTROL

```
%%%%% Input Variables

% Surface
cap desired   [0.90;0.90];
xd   [4000;4000];
yd   [2000; 2000];


% Sub Surface
cap desired U   [0.95; 0.95];
xd U   [3800;3800];
yd U   [2500; 2500];
zd U   [ 800;  900];

% Global variables
res req   3;                % Required pixels for detection of Small Boat via EO
height   4.5;
horizon   (sqrt(17*height)+sqrt(17*height)) * 1000;

pfa   0.01;  % Assuming Probablity of False Alarm as 0.01
sub 1   erfcinv(2*pfa);
f   150;                                % Frequency of Sonar
alpha   (0.036*f^2)/(f^2 + 3600)+(3.2 * 10^ 7 * f^2);

% Obstacles

x1 obs   563.2095; % 100s for 1
y1 obs     223.6816;

x2 obs    3.2218e+03; % 400 for 2
y2 obs    165.5937;

x3 obs     2.7298e+03; % 300 for 3
y3 obs    131.4911;

obs mat   [x1 obs;y1 obs;x2 obs;y2 obs;x3 obs;y3 obs]



x1 obs u   970.0342; % 45 for 1
y1 obs u   167.8621;
z1 obs u      231.2690;

x2 obs u        3.6648e+03; %200 for USV 3
y2 obs u    3.6648e+03;
z2 obs u     836.6365;


obs mat u   [x1 obs u;y1 obs u;z1 obs u;x2 obs u;y2 obs u;z2 obs u;]
vobst u   [0;0;0;0;0;0;0;0;0]
vunit u   [0;0;0;0;0;0;0;0;0]


% Create USVs
x1   150.2;
y1   150.2;
x2   150.1;
y2   150.1;
x3   150;
```

```
y3   150;
x4   149.999;
y4   149.999;

swarmstate   [x1;y1;x2;y2;x3;y3;x4;y4];
x1 plot(1)   x1;
y1 plot(1)   y1;
x2 plot(1)   x2;
y2 plot(1)   y2;
x3 plot(1)   x3;
y3 plot(1)   y3;
x4 plot(1)   x4;
y4 plot(1)   y4;
syms c symbol x1s y1s x2s y2s x3s y3s x4s y4s
ss   [x1s;y1s;x2s;y2s;x3s;y3s;x4s;y4s];



x1 U   149;
y1 U   149;
z1 U    1;
x2 U   148.5;
y2 U   148.5;
z2 U    1.4;
x3 U   149.5;
y3 U   149.5;
z3 U    1.8;
swarmstate U   [x1 U;y1 U;z1 U;x2 U;y2 U;z2 U; x3 U;y3 U;z3 U];
syms snr s magnitude s TL s x1u y1u z1u x2u y2u z2u x3u y3u z3u
ssu   [x1u;y1u;z1u;x2u;y2u;z2u;x3u;y3u;z3u];
x1 plot u(1)   x1 U;
y1 plot u(1)   y1 U;
z1 plot u(1)   z1 U;
x2 plot u(1)   x2 U;
y2 plot u(1)   y2 U;
z2 plot u(1)   z2 U;
x3 plot u(1)   x3 U;
y3 plot u(1)   y3 U;
z3 plot u(1)   z3 U;



vobst u   zeros(size(ssu));
vunit u   zeros(size(ssu));
vunit u xdomain   zeros(size(ssu));
vobst   zeros(size(ss));
vunit   zeros(size(ss));


for t   1:1:900    % t   time in seconds
    for i   1:1:length(cap desired U)
        for n   1:1:((length(swarmstate U))/3)    % n   vessel number
            R   sqrt((xd U(i)   (swarmstate U(n*3   2)))^2 + (yd U(i)   swarmstate U(n*3   1))^2
+ (zd U(i)   swarmstate U(n*3))^2);
            R s   sqrt((xd U(i)   (ssu(n*3   2)))^2 + (yd U(i)   ssu(n*3   1))^2 + (zd U(i)
ssu(n*3))^2);
            if R > 501
                pd(n)   1/R;
                pd symbol(n)   1/R s;
            else

                TL   10*log(R) + 30 + alpha*R;        % Transmission Loss based on Cylindrical
Spreading
                TS   10*log(0.25*10^2);               % Target Strength based on large sphere w/
10m radius
                SL   180;                             % Assuming Sonar Source Level of 180db
                snr   SL   2*TL + TS;
                magnitude   10^(snr/10);
                fhandle   @(x) exp( x.^2);
```

```
                pd(n)   0.5 * (2/sqrt(pi))*integral(fhandle,sub 1 sqrt(magnitude),Inf);


                TL s    10*log(R s) + 30 + alpha*R s;
                snr s   SL   2*TL s + TS;
                magnitude s   10^(snr s/10);
                pd symbol(n)   0.5 * (2/sqrt(pi))*int(fhandle,sub 1 sqrt(magnitude s),Inf);
            end

        end
        % calculate overall capability
        capability u(i)   0;
        cadd u   1;
        for o   1:1:((length(swarmstate U))/3)
            capability u(i)   capability u(i) + pd(o)*cadd u;
            cadd u   (1   capability u(i));
        end

        overall cap u 0;
        cadd   1;
        for o   1:1:((length(swarmstate U))/3)
            overall cap u   overall cap u + pd symbol(o)*cadd;
            cadd   (1   overall cap u);
        end

        jacobian s u   jacobian(overall cap u,[x1u,y1u,z1u,x2u,y2u,z2u,x3u,y3u,z3u]);       %
calculate jacobian matrix based on partial derivative of position 0f (x1, y1)...
        jacobian subbed u
subs(jacobian s u,[x1u,y1u,z1u,x2u,y2u,z2u,x3u,y3u,z3u],[swarmstate U(1),swarmstate U(2),swarmsta
te U(3),swarmstate U(4),swarmstate U(5),swarmstate U(6),swarmstate U(7),swarmstate U(8),swarmstat
e U(9)]);
        jaco mat u(i,:)   (double(jacobian subbed u)); % final jacobian in numbers

    end

    % ADD IN OBSTACLE AVOIDANCE

     safezone u   100;

    for n   1:1:((length(swarmstate U))/3)    % n   vessel number
        objrepx u   0;
        objrepy u   0;
        objrepz u   0;
        for u   1:1:((length(obs mat u)/3))   % obs   number of obstructions
            dist u   sqrt((swarmstate U(n*3   2) obs mat u(u*3 2))^2 + (swarmstate U(n*3   1)
obs mat u(u*3  1))^2 + (swarmstate U(n*3) obs mat u(u*3))^2);

            if dist u < safezone u
                angle u alpha   atan2((swarmstate U((n*3 2))   obs mat u((u*3
2))),(swarmstate U((n*3) 1)   obs mat u(u*3 1)));
                angle u beta   atan2((swarmstate U((n*3))
obs mat u((u*3))),(swarmstate U((n*3) 1)   obs mat u(u*3 1)));
                angle u charlie   atan2((swarmstate U((n*3))
obs mat u((u*3))),(swarmstate U((n*3) 2)   obs mat u(u*3 2)));

                objrepx u   objrepx u + 300*((1/dist u) (1/safezone u))*cos(angle u alpha) +
300*((1/dist u) (1/safezone u))*cos(angle u charlie);
                objrepy u   objrepy u + 300*((1/dist u) (1/safezone u))*sin(angle u alpha) +
300*((1/dist u) (1/safezone u))*cos(angle u beta);
                objrepz u   objrepz u + 300*((1/dist u) (1/safezone u))*sin(angle u beta) +
300*((1/dist u) (1/safezone u))*sin(angle u charlie);
            end

        end
        vobst u((n*3) 2)  objrepx u;
        vobst u(n*3  1)  objrepy u;
        vobst u(n*3)  objrepz u;
```

```
    end

    % ADD IN UNIT AVOIDANCE

    safezone unit u   100;

    for n   1:1:((length(swarmstate U))/3)    % n   vessel number
        unitrepx u   0;
        unitrepy u   0;
        unitrepz u   0;
        for o   1:1:((length(swarmstate U))/3)    % o   number of vessels

            if n    o
                vunit u((n*3) 2)  0;
                vunit u((n*3) 1)  0;
                vunit u(n*3)  0;
            else
                dist unit u   sqrt((swarmstate U(n*3   2) swarmstate U(o*3   2))^2 +
(swarmstate U(n*3   1) swarmstate U(o*3   1))^2 + (swarmstate U(n*3) swarmstate U(o*3))^2);

                if dist unit u < safezone unit u

                    angle unit u alpha   atan2(((swarmstate U((n*3 2))   swarmstate U((o*3
2)))),((swarmstate U((n*3) 1)   swarmstate U(o*3 1))));
                    angle unit u beta   atan2(((swarmstate U((n*3))
swarmstate U((o*3)))),((swarmstate U((n*3) 1)   swarmstate U(o*3 1))));
                    angle unit u charlie   atan2(((swarmstate U((n*3))
swarmstate U((o*3)))),((swarmstate U((n*3) 2)   swarmstate U(o*3 2))));


                    unitrepx u   unitrepx u + 1200*((1/dist unit u)
(1/safezone unit u))*cos(angle unit u alpha)+ 1200*((1/dist unit u)
(1/safezone unit u))*cos(angle unit u charlie);
                    unitrepy u   unitrepy u + 1200*((1/dist unit u)
(1/safezone unit u))*sin(angle unit u alpha)+ 1200*((1/dist unit u)
(1/safezone unit u))*cos(angle unit u beta);
                    unitrepz u   unitrepz u + 1200*((1/dist unit u)
(1/safezone unit u))*sin(angle unit u beta)+ 1200*((1/dist unit u)
(1/safezone unit u))*sin(angle unit u charlie);

                end

            end
            vunit u((n*3) 2)  unitrepx u;
            vunit u((n*3)  1)  unitrepy u;
            vunit u(n*3)  unitrepz u;


        end
    end


    % ADD IN SURFACE / SUBSURFACEUNIT AVOIDANCE

    safezone xdomain unit u   100;

    for n   1:1:((length(swarmstate U))/3)    % n   vessel number
        unitrepx xdomain u   0;
        unitrepy xdomain u   0;
        unitrepz xdomain u   0;
        for o   1:1:((length(swarmstate))/2)    % o   number of vessels
            dist unit xdomain u   sqrt((swarmstate U(n*3   2) swarmstate(o*2   1))^2 +
(swarmstate U(n*3   1) swarmstate U(n*2))^2 + (swarmstate U(n*3) 0)^2);

            if dist unit xdomain u < safezone xdomain unit u
```

```
                angle unit u alpha xdomain   atan2(((swarmstate U((n*3 2))   swarmstate U((o*2
1)))),((swarmstate U((n*3) 1)   swarmstate U(o*2))));
                angle unit u beta xdomain   atan2(((swarmstate U((n*3))
0)),((swarmstate U((n*3) 1)   swarmstate U(o*2))));
                angle unit u charlie xdomain   atan2(((swarmstate U((n*3))
0)),((swarmstate U((n*3) 2)   swarmstate U(o*2 1))));


                unitrepx xdomain u   unitrepx xdomain u + 1200*((1/dist unit xdomain u)
(1/safezone xdomain unit u))*cos(angle unit u alpha xdomain)+ 1200*((1/dist unit xdomain u)
(1/safezone xdomain unit u))*cos(angle unit u charlie xdomain);
                unitrepy xdomain u   unitrepy xdomain u + 1200*((1/dist unit xdomain u)
(1/safezone xdomain unit u))*sin(angle unit u alpha xdomain)+ 1200*((1/dist unit xdomain u)
(1/safezone xdomain unit u))*cos(angle unit u beta xdomain);
                unitrepz xdomain u   unitrepz xdomain u + 1200*((1/dist unit xdomain u)
(1/safezone xdomain unit u))*sin(angle unit u beta xdomain)+ 1200*((1/dist unit xdomain u)
(1/safezone xdomain unit u))*sin(angle unit u charlie xdomain);

            end

        end
        vunit u xdomain((n*3) 2)  unitrepx xdomain u;
        vunit u xdomain((n*3)  1)  unitrepy xdomain u;
        vunit u xdomain(n*3)  unitrepz xdomain u;

    end

    % CONTROL LAW FOR UNDERWATER
   jacob pinv u   pinv(jaco mat u);
    capability u t  transpose(capability u);
    capability plot u(t,:)   capability u;
    cdot u   0.09*(cap desired U   capability u t);
    [sizex u, sizey u]   size(jacob pinv u*jaco mat u);

    swarmstatedot u   jacob pinv u * cdot u;
%     + 45000*((eye(sizex u,sizey u)   jacob pinv u*jaco mat u)*vobst u) +
45000*((eye(sizex u,sizey u)   jacob pinv u*jaco mat u)*vunit u) + 45000*((eye(sizex u,sizey u)
jacob pinv u*jaco mat u)*vunit u xdomain);

    % Speed Limitations

    for n   1:1:((length(swarmstate U))/3)
        while abs(sqrt(((swarmstatedot u((n*3) 2))^2) + (swarmstatedot u((n*3) 1)^2) +
swarmstatedot u(n*3)^2)) > 30
            if swarmstatedot u((n*3) 2) >  0
                swarmstatedot u((n*3) 2)   0.5*swarmstatedot u((n*3) 2);            else
                swarmstatedot u((n*3) 2)   0.5*swarmstatedot u((n*3) 2);
            end

            if swarmstatedot u((n*3) 1) >  0
                swarmstatedot u((n*3) 1)   0.5*swarmstatedot u((n*3) 1);
            else
                swarmstatedot u((n*3) 1)   0.5*swarmstatedot u((n*3) 1);
            end

            if swarmstatedot u(n*3) >  0
                swarmstatedot u(n*3)   0.5*swarmstatedot u(n*3);
            else
                swarmstatedot u(n*3)   0.5*swarmstatedot u(n*3);
            end
        end
    end

    swarmstate U   swarmstate U + swarmstatedot u;
    for n   1:1:((length(swarmstate U))/3)    % n   vessel number
        if swarmstate U(n*3) > 0
            swarmstate U(n*3)   0;
        end
    end
```

```
    x1 plot u(t+1)   swarmstate U(1);
    y1 plot u(t+1)   swarmstate U(2);
    z1 plot u(t+1)   swarmstate U(3);
    x2 plot u(t+1)   swarmstate U(4);
    y2 plot u(t+1)   swarmstate U(5);
    z2 plot u(t+1)   swarmstate U(6);
    x3 plot u(t+1)   swarmstate U(7);
    y3 plot u(t+1)   swarmstate U(7);
    z3 plot u(t+1)   swarmstate U(9);

    capability UNDERWATER(t)   capability u(1);
    capability UNDERWATER 2(t)   capability u(2);



    %%%%%%%%%%%% SURFACE VESSELS %%%%%%%%%%%%%%%%%%%%%

    for i   1:1:length(cap desired)           % i   amount of capabilities desired

        % CALCULATE CAPABILITIES FOR EACH INDIVIDUAL VESSELS FOR SPECIFIC POINT
        for n   1:1:((length(swarmstate))/2)    % n   vessel number
            if sqrt((yd(i)   swarmstate(n*2))^2 + (xd(i)   swarmstate(((n*2) 1)))^2) > 150000
                c(n)   1/(sqrt((yd(i)   swarmstate(n*2))^2 + (xd(i)   swarmstate(((n*2) 1)))^2));
                c symbol(n)    1 /sqrt((yd(i)   ss(n*2))^2 + (xd(i)   ss(((n*2) 1))^2));

            else
                if ((640/(2*sqrt((yd(i)   swarmstate(n*2))^2 + (xd(i)   swarmstate(((n*2)
1)))^2)*tand(24/2))) > 3)
                    c(n)    1   0.00000001*sqrt((yd(i)   swarmstate(n*2))^2 + (xd(i)
swarmstate(((n*2) 1))^2));

                    c symbol(n)    1   0.00000001*sqrt((yd(i)   ss(n*2))^2 + (xd(i)   ss(((n*2)
1))^2));
                elseif sqrt(((yd(i)   swarmstate(n*2))^2) + ((xd(i)   swarmstate(((n*2) 1)))^2))
> horizon
                    c(n)   1/(sqrt(((yd(i)   swarmstate(n*2))^2) + ((xd(i)   swarmstate(((n*2)
1)))^2)));

                    c symbol(n)   1/(sqrt(((yd(i)   ss(n*2))^2) + ((xd(i)   ss(((n*2) 1)))^2)));
                else
                    c(n)   sin((pi/2)*(640/(2*sqrt((yd(i)   swarmstate(n*2))^2 + (xd(i)
swarmstate(((n*2) 1)))^2)*tand(24/2)))/(3));

                    c symbol(n)   sin((pi/2)*(640/(2*sqrt((yd(i)   ss(n*2))^2 + (xd(i)
ss(((n*2) 1)))^2)*tand(24/2)))/(3));
                end
            end
        end


        % CALCULATE OVERALL CAPABILITY
        capability(i)   0 ;
        cadd   1;
        for o   1:1:((length(swarmstate))/2)
            capability(i)   capability(i) + c(o)*cadd;
            cadd   (1   capability(i));
        end

        overall cap   0 ;
        cadd   1;
        for o   1:1:((length(swarmstate))/2)
            overall cap   overall cap + c symbol(o)*cadd;
            cadd   (1   overall cap);
        end


        % SUB CAPABILITY(SYMBOLS) INTO JACOBIAN
```

```
        jacobian s   jacobian(overall cap,[x1s,y1s,x2s,y2s,x3s,y3s,x4s,y4s]);       % calculate
jacobian matrix based on partial derivative of position 0f (x1, y1)...
        jacobian subbed
subs(jacobian s,[x1s,y1s,x2s,y2s,x3s,y3s,x4s,y4s],[swarmstate(1),swarmstate(2),swarmstate(3),swar
mstate(4),swarmstate(5),swarmstate(6),swarmstate(7),swarmstate(8)]);
        jaco mat(i,:)   (double(jacobian subbed)); % final jacobian in numbers

    end

    % ADD IN OBSTACLE AVOIDANCE

    safezone   100;

    for n   1:1:((length(swarmstate))/2)    % n   vessel number
        objrepx   0;
        objrepy   0;
        for obs   1:1:((length(obs mat)/2))   % obs   number of obstructions
            dist   sqrt((swarmstate((n*2) 1)   obs mat((obs*2) 1))^2 + (swarmstate((n*2))
obs mat((obs*2)))^2);

            if dist < safezone
                angle   atan2((swarmstate((n*2))   obs mat((obs*2))),(swarmstate((n*2) 1)
obs mat((obs*2) 1)));
                objrepx   objrepx + 300*((1/dist) (1/safezone))*cos(angle);
                objrepy   objrepy + 300*((1/dist) (1/safezone))*sin(angle);
            end
            vobst rec(:,t)   vobst;

        end
        vobst((n*2) 1)  objrepx;
        vobst(n*2)  objrepy;

    end

    % ADD IN UNIT AVOIDANCE

    safezone unit   100;

    for n   1:1:((length(swarmstate))/2)    % n   vessel number
        unitrepx   0;
        unitrepy   0;
        for o   1:1:((length(swarmstate))/2)    % o   number of vessels

            if n    o
                vunit((n*2) 1)  0;
                vunit(n*2)  0;
            else
                dist unit   sqrt((swarmstate((n*2) 1)   swarmstate((o*2) 1))^2 +
(swarmstate((n*2))   swarmstate((o*2)))^2);

                if dist unit < safezone unit
                    angle unit   atan2((swarmstate((n*2))   swarmstate(o*2)),(swarmstate((n*2) 1)
  swarmstate((o*2) 1)));
                    unitrepx   unitrepx + 800*((1/dist unit) (1/safezone unit))*cos(angle unit);
                    unitrepy   unitrepy + 800*((1/dist unit) (1/safezone unit))*sin(angle unit);
                end
            end
            vunit rec(:,t)   vunit;

        end
        vunit((n*2) 1)  unitrepx;
        vunit(n*2)  unitrepy;

    end


    % CONTROL LAW
    jacob pinv   pinv(jaco mat);
```

```
    capability t   transpose(capability);
    capability plot(t,:)   capability;
    cdot   0.009*(cap desired   capability t);
    [sizex, sizey]   size(jacob pinv*jaco mat);
    swarmstatedot   jacob pinv*cdot;
%     +150*((eye(sizex,sizey)   jacob pinv*jaco mat)*vobst) + 150*((eye(sizex,sizey)
jacob pinv*jaco mat)*vunit);            % vobst needs to be a 6 by 1 matrix


    % LIMITATIONS FOR SPEED
    for n   1:1:((length(swarmstate))/2)
        while abs(sqrt(((swarmstatedot((n*2) 1))^2) + (swarmstatedot(n*2)^2))) > 30
            if swarmstatedot((n*2) 1) >  0
                swarmstatedot((n*2) 1)   0.1*swarmstatedot((n*2) 1);             % was previous
50 and  20
            else
                swarmstatedot((n*2) 1)   0.1*swarmstatedot((n*2) 1);
            end

            if swarmstatedot(n*2) >  0
                swarmstatedot(n*2)   0.1*swarmstatedot(n*2);
            else
                swarmstatedot(n*2)   0.1*swarmstatedot(n*2);
            end
        end
    end
         capability SURFACE(t)   capability(1);
     capability SURFACE 2(t)   capability(2);

    % CONTROL LAW (CONT'D)
    swarmstate   swarmstate + swarmstatedot;

    x1 plot(t+1)   swarmstate(1);
    y1 plot(t+1)   swarmstate(2);
    x2 plot(t+1)   swarmstate(3);
    y2 plot(t+1)   swarmstate(4);
    x3 plot(t+1)   swarmstate(5);
    y3 plot(t+1)   swarmstate(6);
    x4 plot(t+1)   swarmstate(7);
    y4 plot(t+1)   swarmstate(8);

end

% PLOT
figure(1)
hold on
grid on

% Sub Surface
plot3(transpose(x1 plot u),transpose(y1 plot u),transpose(z1 plot u),'*','LineWidth',2.5,'Color',
[0 0.33 0])
plot3(transpose(x2 plot u),transpose(y2 plot u),transpose(z2 plot u),'*','LineWidth',2.5,'Color',
[0 0.66 0])
plot3(transpose(x3 plot u),transpose(y3 plot u),transpose(z3 plot u),'*','LineWidth',2.5,'Color',
[0 1.0 0])

plot3(xd U,yd U,zd U,'k^','LineWidth',2.0)
plot3(x1 obs u,y1 obs u,z1 obs u,'rd','LineWidth',6.0)
plot3(x2 obs u,y2 obs u,z2 obs u,'rd','LineWidth',6.0)


xlabel('X axis')
ylabel('Y axis')
zlabel('Z axis')

% Surface
plot(x1 plot,y1 plot,'o','Color',[0 0 0.5])
plot(x2 plot,y2 plot,'o','Color',[0 0 1.0])
plot(x3 plot,y3 plot,'o','Color',[0.5 0 0])
```

```
plot(x4 plot,y4 plot,'o','Color',[1.0 0 0])
plot(x1 obs,y1 obs,'rd','LineWidth',6.5)
plot(x2 obs,y2 obs,'rd','LineWidth',6.5)
plot(x3 obs,y3 obs,'rd','LineWidth',6.5)

plot(xd,yd,'^k')
xlabel('X Axis')
ylabel('Y Axis')


figure(2)
hold on
grid on
plot(1:length(capability UNDERWATER),capability UNDERWATER,'m .')
plot(1:length(capability UNDERWATER 2),capability UNDERWATER 2,'m .')
plot(1:length(capability SURFACE),capability SURFACE,'k .')
plot(1:length(capability SURFACE 2),capability SURFACE 2,'k .')
plot([1 length(capability)],[0.95 0.95],'m','LineWidth',3.0)
plot([1 length(capability)],[0.9 0.9],'k','LineWidth',3.0)
xlabel('Time (s)')
ylabel('Capability Delivered')
title('Capability Delivered over Time')
legend('Capability Delivered at Target 1','Capability Delivered at Target 2', 'Capability
Desired')
```

APPENDIX E – CODE FOR COOPERATIVE NAVIGATION

```
%%%% INPUT VARIABLES

% Defining All Symbols Used

syms x1s y1s c symbol snr s magnitude s TL s x1u y1u z1u x2u y2u z2u x3u y3u z3u x4u y4u z4u;

% Surface Capability Desired

cap desired = [0.95;];
xd = [1000;];
yd = [0;];

% Sub Surface Capability Desired

cap desired U = [0.95;0.95];
xd U = [2000;2000];
yd U = [200;1500];
zd U = [ 200; 180];


% Obstacles

x1 obs = 563.2095; % 100s for 1
y1 obs = 223.6816;

x2 obs = 600;
y2 obs = 500;

obs mat = [x1 obs;y1 obs;x2 obs;y2 obs;]


x1 obs u = 800.0342; % 45 for 1
y1 obs u = 100.8621;
z1 obs u =    231.2690;

obs mat u = [x1 obs u;y1 obs u;z1 obs u];


% Creating USVs

x1 = 150.1;
y1 = 150.1;

swarmstate = [x1;y1];

ss = [x1s;y1s];

x1 plot(1) = x1;
y1 plot(1) = y1;

% Creating UUVs

x1 U = 149;
y1 U = 149;
z1 U =  1;
x2 U = 145;
y2 U = 145;
z2 U =  1.4;
x3 U = 155;
y3 U = 155;
z3 U =  1.5;
x4 U = 160;
y4 U = 160;
z4 U =  1.2;

swarmstate U = [x1 U;y1 U;z1 U;x2 U;y2 U;z2 U; x3 U; y3 U; z3 U; x4 U; y4 U; z4 U];
ssu = [x1u;y1u;z1u;x2u;y2u;z2u;x3u;y3u;z3u;x4u;y4u;z4u];
x1 plot u(1) = x1 U;
```

```
y1 plot u(1) = y1 U;
z1 plot u(1) = z1 U;
x2 plot u(1) = x2 U;
y2 plot u(1) = y2 U;
z2 plot u(1) = z2 U;
x3 plot u(1) = x3 U;
y3 plot u(1) = y3 U;
z3 plot u(1) = z3 U;
x4 plot u(1) = x4 U;
y4 plot u(1) = y4 U;
z4 plot u(1) = z4 U;


% Etc.

res req = 3;                % Required pixels for detection of Small Boat via EO
height = 4.5;
horizon = (sqrt(17*height)+sqrt(17*height)) * 1000;

pfa = 0.01;  % Assuming Probablity of False Alarm as 0.01
sub 1 = erfcinv(2*pfa);
f = 150;                                % Frequency of Sonar
alpha = (0.036*f^2)/(f^2 + 3600)+(3.2 * 10^ 7 * f^2);

detect = 0;
x target = 3100;
y target = 1500;


vunit u attloc = zeros(size(ssu));
vunit u attloc surf = zeros(size(ss));
vobst u = zeros(size(ssu));
vunit u = zeros(size(ssu));
vunit u xdomain = zeros(size(ssu));
vobst = zeros(size(ss));
vunit = zeros(size(ss));
time = [0, 0, 0, 0];
capability worstcase = 0;

h =
plot3(transpose(x1 plot u(1)),transpose(y1 plot u(1)),transpose(z1 plot u(1)),'*','LineWidth',2.5,'Color',[0
0.33 0]);
h1 =
plot3(transpose(x2 plot u(1)),transpose(y2 plot u(1)),transpose(z2 plot u(1)),'*','LineWidth',2.5,'Color',[0
0.66 0]);
h2 = plot(x1 plot(1),y1 plot(1),'o','Color',[0 0 0.5]);
h3 =
plot3(transpose(x3 plot u(1)),transpose(y3 plot u(1)),transpose(z3 plot u(1)),'*','LineWidth',2.5,'Color',[0
0.99 0]);
h4 =
plot3(transpose(x4 plot u(1)),transpose(y4 plot u(1)),transpose(z4 plot u(1)),'*','LineWidth',2.5,'Color',[0
0.25 0.25]);
h5 = plot(x1 plot u(1),y1 plot u(1));
h6 = plot(x1 plot u(1),y1 plot u(1));
h7 = plot(x1 plot u(1),y1 plot u(1));
h8 = plot(x1 plot u(1),y1 plot u(1));
text1 = text(xd U(1),yd U(1),zd U(1),' ');
text2 = text(xd U(1),yd U(1),zd U(1),' ');
text3 = text(xd U(1),yd U(1),zd U(1),' ');
text4 = text(xd U(1),yd U(1),zd U(1),' ');
text5 = text(xd U(1),yd U(1),zd U(1),' ');
text6 = text(xd U(1),yd U(1),zd U(1),' ');
text7 = text(xd U(1),yd U(1),zd U(1),' ');
text8 = text(xd U(1),yd U(1),zd U(1),' ');
text9 = text(xd U(1),yd U(1),zd U(1),' ');
text10 = text(xd U(1),yd U(1),zd U(1),' ');

%%%%%%%% START TIME AND CONTROL LOOP %%%%%%%%%%

for t = 1:1:10000

    %%%%% SURFACE VESSELS CONTROL LOOP %%%%%%%

    for i = 1:1:length(cap desired)
        % CALCULATE CAPABILITIES FOR EACH INDIVIDUAL VESSELS FOR SPECIFIC POINT
        for n = 1:1:((length(swarmstate))/2)
            if sqrt((yd(i)   swarmstate(n*2))^2 + (xd(i)   swarmstate(((n*2) 1)))^2) > 150000
                c(n) = 1/(sqrt((yd(i)   swarmstate(n*2))^2 + (xd(i)   swarmstate(((n*2) 1)))^2));
                c symbol(n) =  1 /sqrt((yd(i)   ss(n*2))^2 + (xd(i)   ss(((n*2) 1))^2));
```

```
            else
                if ((640/(2*sqrt((yd(i)   swarmstate(n*2))^2 + (xd(i)   swarmstate(((n*2) 1)))^2)*tand(24/2)))
> 3)
                    c(n) =  1   0.00000001*sqrt((yd(i)   swarmstate(n*2))^2 + (xd(i)   swarmstate(((n*2)
1))^2));

                    c symbol(n) =  1   0.00000001*sqrt((yd(i)   ss(n*2))^2 + (xd(i)   ss(((n*2) 1))^2));
                elseif sqrt(((yd(i)   swarmstate(n*2))^2) + ((xd(i)   swarmstate(((n*2) 1)))^2)) > horizon
                    c(n) = 1/(sqrt(((yd(i)   swarmstate(n*2))^2) + ((xd(i)   swarmstate(((n*2) 1)))^2)));

                    c symbol(n) = 1/(sqrt(((yd(i)   ss(n*2))^2) + ((xd(i)   ss(((n*2) 1)))^2)));
                else
                    c(n) = sin((pi/2)*(640/(2*sqrt((yd(i)   swarmstate(n*2))^2 + (xd(i)   swarmstate(((n*2)
1)))^2)*tand(24/2)))/(3));

                    c symbol(n) = sin((pi/2)*(640/(2*sqrt((yd(i)   ss(n*2))^2 + (xd(i)   ss(((n*2)
1)))^2)*tand(24/2)))/(3));
                end
            end
        end


        % CALCULATE OVERALL CAPABILITY
        capability(i) = 0 ;
        cadd = 1;
        for o = 1:1:((length(swarmstate))/2)
            capability(i) = capability(i) + c(o)*cadd;
            cadd = (1   capability(i));
        end

        overall cap = 0 ;
        cadd = 1;
        for o = 1:1:((length(swarmstate))/2)
            overall cap = overall cap + c symbol(o)*cadd;
            cadd = (1   overall cap);
        end


        % SUB CAPABILITY(SYMBOLS) INTO JACOBIAN
        jacobian s = jacobian(overall cap,[x1s,y1s]);       % calculate jacobian matrix based on partial
derivative of position 0f (x1, y1)...
        jacobian subbed = subs(jacobian s,[x1s,y1s],[swarmstate(1),swarmstate(2)]);
        jaco mat(i,:) = (double(jacobian subbed)); % final jacobian in numbers

    end

    % ALL SECONDARY CONTROL VARAIBLES DEFINED

    safezone = 100;
    safezone unit = 20;

    % SECONDARY CONTROL CODE

    for n = 1:1:((length(swarmstate))/2)

        % ASSISTING LOCALIZATION
        unitattx surf = 0;
        unitatty surf = 0;
        for nol = 1:1:((length(swarmstate U))/3)

            if time(nol) > 300

                angle unit u alpha att = atan2(((swarmstate U((nol*3 2))   swarmstate((n*2
1))),((swarmstate U((nol*3) 1)   swarmstate(n*2))));


                unitattx surf = unitattx surf + 500*((500/10))*sin(angle unit u alpha att);
                unitatty surf = unitatty surf + 500*((500/10))*cos(angle unit u alpha att);
            else
                unitattx surf = 0;
                unitatty surf = 0;
            end
        end

        vunit u attloc surf((n*2) 1) = unitattx surf;
        vunit u attloc surf((n*2)) = unitatty surf;
```

```
    end

    % CONTROL LAW FOR SURFACE
    jacob pinv = pinv(jaco mat);
    capability t = transpose(capability);
    capability plot(t,:) = capability;
    cdot = 9*(cap desired   capability t);
    [sizex, sizey] = size(jacob pinv*jaco mat);
    swarmstatedot = 10*jacob pinv*cdot +150*((eye(sizex,sizey)   jacob pinv*jaco mat)*vobst) +
150*((eye(sizex,sizey)   jacob pinv*jaco mat)*vunit) + 0.001*((eye(sizex,sizey)
jacob pinv*jaco mat)*vunit u attloc surf);             % vobst needs to be a 6 by 1 matrix
    hi = 150*((eye(sizex,sizey)   jacob pinv*jaco mat)*vobst);

    % LIMITATIONS FOR SPEED
    for n = 1:1:((length(swarmstate))/2)
        while abs(sqrt(((swarmstatedot((n*2) 1))^2) + (swarmstatedot(n*2)^2))) > 10.5
            if swarmstatedot((n*2) 1) >= 0
                swarmstatedot((n*2) 1) = 0.1*swarmstatedot((n*2) 1);
            else
                swarmstatedot((n*2) 1) = 0.1*swarmstatedot((n*2) 1);
            end

            if swarmstatedot(n*2) >= 0
                swarmstatedot(n*2) = 0.1*swarmstatedot(n*2);
            else
                swarmstatedot(n*2) = 0.1*swarmstatedot(n*2);
            end
        end
    end
    capability SURFACE(t) = capability(1);

    % CONTROL LAW (CONT'D)
    swarmstate = swarmstate + swarmstatedot;

    x1 plot(t+1) = swarmstate(1);
    y1 plot(t+1) = swarmstate(2);

    %%%%%%%%%%%%%% CONTROL LOOP FOR UNDERWATER VEHICLES %%%%%%%%%%%%%%%
    for i = 1:1:length(cap desired U)
        for n = 1:1:((length(swarmstate U))/3)    % n = vessel number
            R = sqrt((xd U(i)   (swarmstate U(n*3   2)))^2 + (yd U(i)   swarmstate U(n*3   1))^2 + (zd U(i)
swarmstate U(n*3))^2);
            R s = sqrt((xd U(i)   (ssu(n*3   2)))^2 + (yd U(i)   ssu(n*3   1))^2 + (zd U(i)   ssu(n*3))^2);
            if R > 501
                pd(n) = 1/R;
                pd symbol(n) = 1/R s;
            else

                TL = 10*log(R) + 30 + alpha*R;        % Transmission Loss based on Cylindrical Spreading
                TS = 10*log(0.25*10^2);               % Target Strength based on large sphere w/ 10m radius
                SL = 180;                             % Assuming Sonar Source Level of 180db
                snr = SL   2*TL + TS;
                magnitude = 10^(snr/10);
                fhandle = @(x) exp( x.^2);
                pd(n) = 0.5 * (2/sqrt(pi))*integral(fhandle,sub 1 sqrt(magnitude),Inf);


                TL s  = 10*log(R s) + 30 + alpha*R s;
                snr s = SL   2*TL s + TS;
                magnitude s = 10^(snr s/10);
                pd symbol(n) = 0.5 * (2/sqrt(pi))*int(fhandle,sub 1 sqrt(magnitude s),Inf);
            end

        end
        % calculate overall capability
        capability u(i) = 0;
        cadd u = 1;
        for o = 1:1:((length(swarmstate U))/3)
            capability u(i) = capability u(i) + pd(o)*cadd u;
            cadd u = (1   capability u(i));
        end

        overall cap u=0;
        cadd = 1;
        for o = 1:1:((length(swarmstate U))/3)
            overall cap u = overall cap u + pd symbol(o)*cadd;
            cadd = (1   overall cap u);
        end
```

```
        jacobian s u = jacobian(overall cap u,[x1u,y1u,z1u,x2u,y2u,z2u,x3u,y3u,z3u,x4u,y4u,z4u]);       %
calculate jacobian matrix based on partial derivative of position 0f (x1, y1)...
        jacobian subbed u =
subs(jacobian s u,[x1u,y1u,z1u,x2u,y2u,z2u,x3u,y3u,z3u,x4u,y4u,z4u],[x1 plot u(t),y1 plot u(t),z1 plot u(t),x2
plot u(t),y2 plot u(t),z2 plot u(t),x3 plot u(t),y3 plot u(t),z3 plot u(t),x4 plot u(t),y4 plot u(t),z4 plot u(
t)]);
        jaco mat u(i,:) = (double(jacobian subbed u)); % final jacobian in numbers
    end

    % ADD IN SECONDARY CONTROLS FOR UNDERWATER VESSELS

    % Defining Variables

    safezone u = 100;
    safezone unit u = 20;
    safezone xdomain unit u = 20;
    unitrepx xdomain u = 0;
    unitrepy xdomain u = 0;
    unitrepz xdomain u = 0;
    objrepx u = 0;
    objrepy u = 0;
    objrepz u = 0;
    unitattx = 0;
    unitatty = 0;
    unitattz = 0;
    unitrepx u = 0;
    unitrepy u = 0;
    unitrepz u = 0;

        for no sub = 1:1:length(swarmstate U)/3

            for no surface = 1:1:length(swarmstate)/2

                distance for localization(no sub) = sqrt((swarmstate U(no sub*3 2) (swarmstate((no surface*2)
1)))^2 + (swarmstate U((no sub*3) 1) swarmstate(no surface*2))^2 + (swarmstate U(no sub*3))^2);
                if distance for localization(no sub) < 500
                    swarmstate U(no sub*3 2) = swarmstate U(no sub*3 2);
                    swarmstate U(no sub*3 1) = swarmstate U(no sub*3 1);
                    swarmstate U(no sub*3) = swarmstate U(no sub*3);
                    time(no sub) = 0;
                else
                    swarmstate U(no sub*3 2) = swarmstate U(no sub*3 2);
                    swarmstate U(no sub*3 1) = swarmstate U(no sub*3 1);
                    swarmstate U(no sub*3) = swarmstate U(no sub*3);
                    time(no sub) = time(no sub) + 1;
                end
            end
        end

        % ASSIT IN LOCALIZATION
        for uol = 1:1:((length(swarmstate))/2)

            [val idx] = max(time);
            if val > 300

                angle unit u alpha att = atan2(((swarmstate U((idx*3 2))   swarmstate((uol*2
1)))),((swarmstate U(idx*3) 1)   swarmstate(uol*2)));
                angle unit u beta att = atan2(((swarmstate U((idx*3))   0)),((swarmstate U((idx*3) 1)
swarmstate(uol*2))));
                angle unit u charlie att = atan2(((swarmstate U((idx*3))   0)),((swarmstate U((idx*3) 2)
swarmstate(uol*2 1))));

                magnitude alpha = norm(swarmstate U((idx*3 2):(idx*3 1))   swarmstate);
                magnitude beta = norm(swarmstate U((idx*3 1):(idx*3))   [swarmstate(2);0]);
                magnitude charlie = norm([swarmstate U(idx*3 2);swarmstate U(idx*3)]   [swarmstate(1);0]);

                unitattx = unitattx   magnitude alpha*sin(angle unit u alpha att)
magnitude charlie*cos(angle unit u charlie att);
                unitatty = unitatty + magnitude alpha*cos(angle unit u alpha att)
magnitude beta*cos(angle unit u beta att);
                unitattz = unitattz   magnitude beta*sin(angle unit u beta att)
magnitude charlie*sin(angle unit u charlie att);

                vunit u attloc = [0;0;0;0;0;0;0;0;0;0;0;0];
                vunit u attloc((idx*3) 2) = unitattx;
                vunit u attloc((idx*3) 1) = unitatty;
                vunit u attloc(idx*3) = unitattz;
```

```
            else
                vunit u attloc = [0;0;0;0;0;0;0;0;0;0;0;0];
            end
        end


    % CONTROL LAW FOR UNDERWATER
    jacob pinv u = pinv(jaco mat u);
    capability u t =transpose(capability worstcase);
    capability plot u(t,:) = capability u;
    cdot u = 1000*(cap desired U   capability u t);
    [sizex u, sizey u] = size(jacob pinv u*jaco mat u);

    swarmstatedot u = 100*jacob pinv u * cdot u + 45000*((eye(sizex u,sizey u)
jacob pinv u*jaco mat u)*vunit u attloc)  +  45*((eye(sizex u,sizey u)   jacob pinv u*jaco mat u)*vobst u) +
45*((eye(sizex u,sizey u)   jacob pinv u*jaco mat u)*vunit u) + 45*((eye(sizex u,sizey u)
jacob pinv u*jaco mat u)*vunit u xdomain);


    % Speed Limitations

    for n = 1:1:((length(swarmstate U))/3)
        while abs(sqrt(((swarmstatedot u((n*3) 2))^2) + (swarmstatedot u((n*3) 1)^2) + swarmstatedot u(n*3)^2))
> 8
            if swarmstatedot u((n*3) 2) >= 0
                swarmstatedot u((n*3) 2) = 0.5*swarmstatedot u((n*3) 2);            else
                swarmstatedot u((n*3) 2) = 0.5*swarmstatedot u((n*3) 2);
            end

            if swarmstatedot u((n*3) 1) >= 0
                swarmstatedot u((n*3) 1) = 0.5*swarmstatedot u((n*3) 1);
            else
                swarmstatedot u((n*3) 1) = 0.5*swarmstatedot u((n*3) 1);
            end

            if swarmstatedot u(n*3) >= 0
                swarmstatedot u(n*3) = 0.5*swarmstatedot u(n*3);
            else
                swarmstatedot u(n*3) = 0.5*swarmstatedot u(n*3);
            end
        end
    end

    % FINAL SWARMSTATE FOR UNDERWATER
    swarmstate U = swarmstate U + swarmstatedot u;
    for n = 1:1:((length(swarmstate U))/3)    % n = vessel number
        if swarmstate U(n*3) > 0
            swarmstate U(n*3) = 0;
        end
    end

    % CALCULATING WORST CASE CAPABILITY

    for n = 1:1:length(swarmstate U)/3
        unc(n) = time(n)*0.160376;
    end

    for n = 1:1:length(xd U)
        if (xd U(n)   swarmstate U(1)) > 0
            x1 plot u(t+1) = swarmstate U(1)   unc(1);
        else
            x1 plot u(t+1) = swarmstate U(1) + unc(1);
        end

        if (yd U(n)   swarmstate U(2)) > 0
            y1 plot u(t+1) = swarmstate U(2)   unc(1);
        else
            y1 plot u(t+1) = swarmstate U(2) + unc(1);
        end

        if (zd U(n)   swarmstate U(3)) > 0
            z1 plot u(t+1) = swarmstate U(3)   unc(1);
        else
            z1 plot u(t+1) = swarmstate U(3) + unc(1);
            if z1 plot u(t+1) > 0
                z1 plot u(t+1) = 0;
```

```
            end
        end

        if (xd U(n)   swarmstate U(4)) > 0
            x2 plot u(t+1) = swarmstate U(4)   unc(2);
        else
            x2 plot u(t+1) = swarmstate U(4) + unc(2);
        end

        if (yd U(n)   swarmstate U(5)) > 0
            y2 plot u(t+1) = swarmstate U(5)   unc(2);
        else
            y2 plot u(t+1) = swarmstate U(5) + unc(2);
        end

        if (zd U(n)   swarmstate U(6)) > 0
            z2 plot u(t+1) = swarmstate U(6)   unc(2);
        else
            z2 plot u(t+1) = swarmstate U(6) + unc(2);
            if z2 plot u(t+1) > 0
                z2 plot u(t+1) = 0;
            end
        end

        if (xd U(n)   swarmstate U(7)) > 0
            x3 plot u(t+1) = swarmstate U(7)   unc(3);
        else
            x3 plot u(t+1) = swarmstate U(7) + unc(3);
        end

        if (yd U(n)   swarmstate U(8)) > 0
            y3 plot u(t+1) = swarmstate U(8)   unc(3);
        else
            y3 plot u(t+1) = swarmstate U(8) + unc(3);
        end

        if (zd U(n)   swarmstate U(9)) > 0
            z3 plot u(t+1) = swarmstate U(9)   unc(3);
        else
            z3 plot u(t+1) = swarmstate U(9) + unc(3);
            if z3 plot u(t+1) > 0
                z3 plot u(t+1) = 0;
            end
        end



        if (xd U(n)   swarmstate U(10)) > 0
            x4 plot u(t+1) = swarmstate U(10)   unc(4);
        else
            x4 plot u(t+1) = swarmstate U(10) + unc(4);
        end

        if (yd U(n)   swarmstate U(11)) > 0
            y4 plot u(t+1) = swarmstate U(11)   unc(4);
        else
            y4 plot u(t+1) = swarmstate U(11) + unc(4);
        end

        if (zd U(n)   swarmstate U(12)) > 0
            z4 plot u(t+1) = swarmstate U(12)   unc(4);
        else
            z4 plot u(t+1) = swarmstate U(12) + unc(4);
            if z4 plot u(t+1) > 0
                z4 plot u(t+1) = 0;
            end
        end

    end


    swarmstate worstcase = [x1 plot u(t+1), y1 plot u(t+1), z1 plot u(t+1), x2 plot u(t+1), y2 plot u(t+1),
z2 plot u(t+1),x3 plot u(t+1),y3 plot u(t+1),z3 plot u(t+1),x4 plot u(t+1),y4 plot u(t+1),z4 plot u(t+1)];



    for n = 1:1:((length(swarmstate U))/3)     % n = vessel number
        if swarmstate worstcase(n*3) > 0
            swarmstate worstcase(n*3) = 0;
```

```
        end
    end

    % USE ABOVE PLOTS TO CALCULATE WORST CASE CAPABILITY DELIVERED

    for i = 1:1:length(cap desired U)

        for n = 1:1:((length(swarmstate U))/3)    % n = vessel number
            R = sqrt((xd U(i)   (swarmstate worstcase(n*3   2)))^2 + (yd U(i)   swarmstate worstcase(n*3
1))^2 + (zd U(i)   swarmstate worstcase(n*3))^2);
            if R > 501
                pd(n) = 1/R;
            else

                TL = 10*log(R) + 30 + alpha*R;        % Transmission Loss based on Cylindrical Spreading
                TS = 10*log(0.25*10^2);               % Target Strength based on large sphere w/ 10m radius
                SL = 180;                             % Assuming Sonar Source Level of 180db
                snr = SL   2*TL + TS;
                magnitude = 10^(snr/10);
                fhandle = @(x) exp( x.^2);
                pd(n) = 0.5 * (2/sqrt(pi))*integral(fhandle,sub 1 sqrt(magnitude),Inf);        %
            end

        end
        % calculate overall capability
        capability worstcase(i) = 0;
        cadd u = 1;
        for o = 1:1:((length(swarmstate U))/3)
            capability worstcase(i) = capability worstcase(i) + pd(o)*cadd u;
            cadd u = (1   capability worstcase(i));
        end
    end


    %%%%%%%%%% PLOTS

    delete(h)
    delete(h1)
    delete(h2)
    delete(h3)
    delete(h4)
    delete(h5)
    delete(h6)
    delete(h7)
    delete(h8)

    delete(text2)
    delete(text4)
    delete(text6)
    delete(text7)
    delete(text8)
    delete(text9)
    delete(text10)

    figure(1)
    axis([0 2500 0 2000  300 0])
    hold on
    grid on
    plot3(xd U,yd U,zd U,'k^','LineWidth',2.0)

    plot(xd,yd,'^k')

    xlabel('X axis')
    ylabel('Y axis')
    zlabel('Z axis')


    h = plot3(swarmstate U(1),swarmstate U(2),swarmstate U(3),'*','LineWidth',2.5,'Color',[0 0.33 0]);
    h1 = plot3(swarmstate U(4),swarmstate U(5),swarmstate U(6),'*','LineWidth',2.5,'Color',[0 0.66 0]);
    h2 = plot(x1 plot(t+1),y1 plot(t+1),'*','Color',[0 0 0.5]);
    h3 = plot3(swarmstate U(7),swarmstate U(8),swarmstate U(9),'*','LineWidth',2.5,'Color',[0 0.99 0]);
    h4 = plot3(swarmstate U(10),swarmstate U(11),swarmstate U(12),'*','LineWidth',2.5,'Color',[0 0.25 0.25]);


    h5 = plot3([x1 plot u(t+1) x1 plot u(t+1)],[y1 plot u(t+1) y1 plot u(t+1)],[z1 plot u(t+1)  300],'
','Color',[0.5 0.5 0.5]);
    h6 = plot3([x2 plot u(t+1) x2 plot u(t+1)],[y2 plot u(t+1) y2 plot u(t+1)],[z2 plot u(t+1)  300],'
','Color',[0.5 0.5 0.5]);
```

```
    h7 = plot3([x3 plot u(t+1) x3 plot u(t+1)],[y3 plot u(t+1) y3 plot u(t+1)],[z3 plot u(t+1)  300],'
','Color',[0.5 0.5 0.5]);
    h8 = plot3([x4 plot u(t+1) x4 plot u(t+1)],[y4 plot u(t+1) y4 plot u(t+1)],[z4 plot u(t+1)  300],'
','Color',[0.5 0.5 0.5]);

    text1 = text(xd, yd, ['Desired Capability:','   ',num2str(cap desired)]);
    text2 = text(xd, yd 50, ['Current Capability:','   ',num2str(capability)]);

    text3 = text(xd U(1), yd U(1), zd U(1), ['Desired Capability:','   ',num2str(cap desired U(1))]);
    text4 = text(xd U(1), yd U(1) 50, zd U(1), ['Current Capability:','   ',num2str(capability worstcase(1))]);

    text5 = text(xd U(2), yd U(2) 100, zd U(2), ['Desired Capability:','   ',num2str(cap desired U(2))]);
    text6 = text(xd U(2), yd U(2) 150, zd U(2), ['Current Capability:','
',num2str(capability worstcase(2))]);

    text7 = text(swarmstate U(1),swarmstate U(2),swarmstate U(3),['TimeLL1:', '  ',num2str(time(1))]);
    text8 = text(swarmstate U(4),swarmstate U(5),swarmstate U(6),['TimeLL2:', '  ',num2str(time(2))]);
    text9 = text(swarmstate U(7),swarmstate U(8),swarmstate U(9),['TimeLL3:', '  ',num2str(time(3))]);
    text10 = text(swarmstate U(10),swarmstate U(11),swarmstate U(12),['TimeLL4:', '  ',num2str(time(4))]);
    title( sprintf( 'Time: %d s', t));

    drawnow
end% END TIME LOOP
```

APPENDIX F – CODE FOR COOPERATIVE NAVIGATION AND INTERDICTION

```
%%%% INPUT VARIABLES

% Defining All Symbols Used

syms x1s y1s c symbol snr s magnitude s TL s x1u y1u z1u x2u y2u z2u x3u y3u z3u x4u y4u z4u;

% Surface Capability Desired

cap desired   [0.95;];
xd   [1000;];
yd   [0;];

% Sub Surface Capability Desired

cap desired U   [0.95;0.95];
xd U   [2000;2000];
yd U   [200;1500];
zd U   [ 200; 180];


% Obstacles

x1 obs   563.2095; % 100s for 1
y1 obs   223.6816;

x2 obs   600;
y2 obs   500;

obs mat   [x1 obs;y1 obs;x2 obs;y2 obs;]


x1 obs u   800.0342; % 45 for 1
y1 obs u   100.8621;
z1 obs u      231.2690;

obs mat u   [x1 obs u;y1 obs u;z1 obs u];


% Creating USVs

x1   150.1;
y1   150.1;

swarmstate   [x1;y1];

ss   [x1s;y1s];

x1 plot(1)   x1;
y1 plot(1)   y1;

% Creating UUVs

x1 U   149;
y1 U   149;
z1 U    1;
x2 U   145;
y2 U   145;
z2 U    1.4;
```

```
x3 U   155;
y3 U   155;
z3 U    1.5;
x4 U   160;
y4 U   160;
z4 U    1.2;

swarmstate U   [x1 U;y1 U;z1 U;x2 U;y2 U;z2 U; x3 U; y3 U; z3 U; x4 U; y4 U; z4 U];
ssu   [x1u;y1u;z1u;x2u;y2u;z2u;x3u;y3u;z3u;x4u;y4u;z4u];
x1 plot u(1)   x1 U;
y1 plot u(1)   y1 U;
z1 plot u(1)   z1 U;
x2 plot u(1)   x2 U;
y2 plot u(1)   y2 U;
z2 plot u(1)   z2 U;
x3 plot u(1)   x3 U;
y3 plot u(1)   y3 U;
z3 plot u(1)   z3 U;
x4 plot u(1)   x4 U;
y4 plot u(1)   y4 U;
z4 plot u(1)   z4 U;


% Etc.

res req   3;                % Required pixels for detection of Small Boat via EO
height   4.5;
horizon   (sqrt(17*height)+sqrt(17*height)) * 1000;

pfa   0.01;  % Assuming Probablity of False Alarm as 0.01
sub 1   erfcinv(2*pfa);
f   150;                                % Frequency of Sonar
alpha   (0.036*f^2)/(f^2 + 3600)+(3.2 * 10^ 7 * f^2);

detect   0;
x target   3100;
y target   1500;



vunit u attloc   zeros(size(ssu));
vunit u attloc surf   zeros(size(ss));
vobst u   zeros(size(ssu));
vunit u   zeros(size(ssu));
vunit u xdomain   zeros(size(ssu));
vobst   zeros(size(ss));
vunit   zeros(size(ss));
time   [0, 0, 0, 0];
capability worstcase   0;

h
plot3(transpose(x1 plot u(1)),transpose(y1 plot u(1)),transpose(z1 plot u(1)),'*','LineWidth',2.5
,'Color',[0 0.33 0]);
h1
plot3(transpose(x2 plot u(1)),transpose(y2 plot u(1)),transpose(z2 plot u(1)),'*','LineWidth',2.5
,'Color',[0 0.66 0]);
h2   plot(x1 plot(1),y1 plot(1),'o','Color',[0 0 0.5]);
h3
plot3(transpose(x3 plot u(1)),transpose(y3 plot u(1)),transpose(z3 plot u(1)),'*','LineWidth',2.5
,'Color',[0 0.99 0]);
h4
plot3(transpose(x4 plot u(1)),transpose(y4 plot u(1)),transpose(z4 plot u(1)),'*','LineWidth',2.5
,'Color',[0 0.25 0.25]);
h5   plot(x1 plot u(1),y1 plot u(1));
h6   plot(x1 plot u(1),y1 plot u(1));
h7   plot(x1 plot u(1),y1 plot u(1));
h8   plot(x1 plot u(1),y1 plot u(1));
text1   text(xd U(1),yd U(1),zd U(1),' ');
text2   text(xd U(1),yd U(1),zd U(1),' ');
text3   text(xd U(1),yd U(1),zd U(1),' ');
text4   text(xd U(1),yd U(1),zd U(1),' ');
```

```
text5   text(xd U(1),yd U(1),zd U(1),' ');
text6   text(xd U(1),yd U(1),zd U(1),' ');
text7   text(xd U(1),yd U(1),zd U(1),' ');
text8   text(xd U(1),yd U(1),zd U(1),' ');
text9   text(xd U(1),yd U(1),zd U(1),' ');
text10   text(xd U(1),yd U(1),zd U(1),' ');

%%%%%%%% START TIME AND CONTROL LOOP %%%%%%%%%%

for t   1:1:10000

    %%%%% SURFACE VESSELS CONTROL LOOP %%%%%%%

    for i   1:1:length(cap desired)
        % CALCULATE CAPABILITIES FOR EACH INDIVIDUAL VESSELS FOR SPECIFIC POINT
        for n   1:1:((length(swarmstate))/2)
            if sqrt((yd(i)   swarmstate(n*2))^2 + (xd(i)   swarmstate(((n*2) 1)))^2) > 150000
                c(n)   1/(sqrt((yd(i)   swarmstate(n*2))^2 + (xd(i)   swarmstate(((n*2) 1)))^2));
                c symbol(n)    1 /sqrt((yd(i)   ss(n*2))^2 + (xd(i)   ss(((n*2) 1))^2));

            else
                if ((640/(2*sqrt((yd(i)   swarmstate(n*2))^2 + (xd(i)   swarmstate(((n*2)
1)))^2)*tand(24/2))) > 3)
                    c(n)    1   0.00000001*sqrt((yd(i)   swarmstate(n*2))^2 + (xd(i)
swarmstate(((n*2) 1))^2));

                    c symbol(n)    1   0.00000001*sqrt((yd(i)   ss(n*2))^2 + (xd(i)   ss(((n*2)
1))^2));
                elseif sqrt(((yd(i)   swarmstate(n*2))^2) + ((xd(i)   swarmstate(((n*2) 1)))^2))
> horizon
                    c(n)   1/(sqrt(((yd(i)   swarmstate(n*2))^2) + ((xd(i)   swarmstate(((n*2)
1)))^2)));

                    c symbol(n)   1/(sqrt(((yd(i)   ss(n*2))^2) + ((xd(i)   ss(((n*2) 1)))^2)));
                else
                    c(n)   sin((pi/2)*(640/(2*sqrt((yd(i)   swarmstate(n*2))^2 + (xd(i)
swarmstate(((n*2) 1)))^2)*tand(24/2)))/(3));

                    c symbol(n)   sin((pi/2)*(640/(2*sqrt((yd(i)   ss(n*2))^2 + (xd(i)
ss(((n*2) 1)))^2)*tand(24/2)))/(3));
                end
            end
        end


        % CALCULATE OVERALL CAPABILITY
        capability(i)   0 ;
        cadd   1;
        for o   1:1:((length(swarmstate))/2)
            capability(i)   capability(i) + c(o)*cadd;
            cadd   (1   capability(i));
        end

        overall cap   0 ;
        cadd   1;
        for o   1:1:((length(swarmstate))/2)
            overall cap   overall cap + c symbol(o)*cadd;
            cadd   (1   overall cap);
        end


        % SUB CAPABILITY(SYMBOLS) INTO JACOBIAN
        jacobian s   jacobian(overall cap,[x1s,y1s]);       % calculate jacobian matrix based on
partial derivative of position 0f (x1, y1)...
        jacobian subbed   subs(jacobian s,[x1s,y1s],[swarmstate(1),swarmstate(2)]);
        jaco mat(i,:)   (double(jacobian subbed)); % final jacobian in numbers

    end
```

```
    % ALL SECONDARY CONTROL VARAIBLES DEFINED

    safezone   100;
    safezone unit   20;

    % SECONDARY CONTROL CODE

    for n   1:1:((length(swarmstate))/2)
        % ASSISTING LOCALIZATION
        unitattx surf   0;
        unitatty surf   0;
        for nol   1:1:((length(swarmstate U))/3)

            if time(nol) > 300

                angle unit u alpha att   atan2(((swarmstate U((nol*3 2))   swarmstate((n*2
1)))),((swarmstate U((nol*3) 1)   swarmstate(n*2))));


                unitattx surf   unitattx surf + 500*((500/10))*sin(angle unit u alpha att);
                unitatty surf   unitatty surf + 500*((500/10))*cos(angle unit u alpha att);
            else
                unitattx surf   0;
                unitatty surf   0;
            end
        end

        vunit u attloc surf((n*2) 1)   unitattx surf;
        vunit u attloc surf((n*2))   unitatty surf;
    end

    % CONTROL LAW FOR SURFACE
    jacob pinv   pinv(jaco mat);
    capability t   transpose(capability);
    capability plot(t,:)   capability;
    cdot   9*(cap desired   capability t);
    [sizex, sizey]   size(jacob pinv*jaco mat);
    swarmstatedot   10*jacob pinv*cdot +150*((eye(sizex,sizey)   jacob pinv*jaco mat)*vobst) +
150*((eye(sizex,sizey)   jacob pinv*jaco mat)*vunit) + 0.001*((eye(sizex,sizey)
jacob pinv*jaco mat)*vunit u attloc surf);            % vobst needs to be a 6 by 1 matrix
    hi   150*((eye(sizex,sizey)   jacob pinv*jaco mat)*vobst);

    % LIMITATIONS FOR SPEED
    for n   1:1:((length(swarmstate))/2)
        while abs(sqrt(((swarmstatedot((n*2) 1))^2) + (swarmstatedot(n*2)^2))) > 10.5
            if swarmstatedot((n*2) 1) >  0
                swarmstatedot((n*2) 1)   0.1*swarmstatedot((n*2) 1);
            else
                swarmstatedot((n*2) 1)   0.1*swarmstatedot((n*2) 1);
            end

            if swarmstatedot(n*2) >  0
                swarmstatedot(n*2)   0.1*swarmstatedot(n*2);
            else
                swarmstatedot(n*2)   0.1*swarmstatedot(n*2);
            end
        end
    end
    capability SURFACE(t)   capability(1);

    % CONTROL LAW (CONT'D)
    swarmstate   swarmstate + swarmstatedot;

    x1 plot(t+1)   swarmstate(1);
    y1 plot(t+1)   swarmstate(2);

    %%%%%%%%%%%%% CONTROL LOOP FOR UNDERWATER VEHICLES %%%%%%%%%%%%%%
```

```
    for i   1:1:length(cap desired U)
        for n   1:1:((length(swarmstate U))/3)    % n   vessel number
            R   sqrt((xd U(i)   (swarmstate U(n*3   2)))^2 + (yd U(i)   swarmstate U(n*3   1))^2
+ (zd U(i)   swarmstate U(n*3))^2);
            R s   sqrt((xd U(i)   (ssu(n*3   2)))^2 + (yd U(i)   ssu(n*3   1))^2 + (zd U(i)
ssu(n*3))^2);
            if R > 501
                pd(n)   1/R;
                pd symbol(n)   1/R s;
            else

                TL   10*log(R) + 30 + alpha*R;        % Transmission Loss based on Cylindrical
Spreading
                TS   10*log(0.25*10^2);               % Target Strength based on large sphere w/
10m radius
                SL   180;                             % Assuming Sonar Source Level of 180db
                snr   SL   2*TL + TS;
                magnitude   10^(snr/10);
                fhandle   @(x) exp( x.^2);
                pd(n)   0.5 * (2/sqrt(pi))*integral(fhandle,sub 1 sqrt(magnitude),Inf);


                TL s    10*log(R s) + 30 + alpha*R s;
                snr s   SL   2*TL s + TS;
                magnitude s   10^(snr s/10);
                pd symbol(n)   0.5 * (2/sqrt(pi))*int(fhandle,sub 1 sqrt(magnitude s),Inf);
            end

        end
        % calculate overall capability
        capability u(i)   0;
        cadd u   1;
        for o   1:1:((length(swarmstate U))/3)
            capability u(i)   capability u(i) + pd(o)*cadd u;
            cadd u   (1   capability u(i));
        end

        overall cap u 0;
        cadd   1;
        for o   1:1:((length(swarmstate U))/3)
            overall cap u   overall cap u + pd symbol(o)*cadd;
            cadd   (1   overall cap u);
        end

        jacobian s u   jacobian(overall cap u,[x1u,y1u,z1u,x2u,y2u,z2u,x3u,y3u,z3u,x4u,y4u,z4u]);
% calculate jacobian matrix based on partial derivative of position 0f (x1, y1)...
        jacobian subbed u
subs(jacobian s u,[x1u,y1u,z1u,x2u,y2u,z2u,x3u,y3u,z3u,x4u,y4u,z4u],[x1 plot u(t),y1 plot u(t),z1
 plot u(t),x2 plot u(t),y2 plot u(t),z2 plot u(t),x3 plot u(t),y3 plot u(t),z3 plot u(t),x4 plot
u(t),y4 plot u(t),z4 plot u(t)]);
        jaco mat u(i,:)   (double(jacobian subbed u)); % final jacobian in numbers
    end

    % ADD IN SECONDARY CONTROLS FOR UNDERWATER VESSELS

    % Defining Variables

    safezone u   100;
    safezone unit u   20;
    safezone xdomain unit u   20;
    unitrepx xdomain u   0;
    unitrepy xdomain u   0;
    unitrepz xdomain u   0;
    objrepx u   0;
    objrepy u   0;
    objrepz u   0;
    unitattx   0;
    unitatty   0;
    unitattz   0;
    unitrepx u   0;
```

```
    unitrepy u   0;
    unitrepz u   0;

        for no sub   1:1:length(swarmstate U)/3

            for no surface   1:1:length(swarmstate)/2

                distance for localization(no sub)   sqrt((swarmstate U(no sub*3 2)
(swarmstate((no surface*2)   1)))^2 + (swarmstate U((no sub*3) 1) swarmstate(no surface*2))^2 +
(swarmstate U(no sub*3))^2);
                if distance for localization(no sub) < 500
                    swarmstate U(no sub*3 2)   swarmstate U(no sub*3 2);
                    swarmstate U(no sub*3 1)   swarmstate U(no sub*3 1);
                    swarmstate U(no sub*3)   swarmstate U(no sub*3);
                    time(no sub)   0;
                else
                    swarmstate U(no sub*3 2)   swarmstate U(no sub*3 2);
                    swarmstate U(no sub*3 1)   swarmstate U(no sub*3 1);
                    swarmstate U(no sub*3)   swarmstate U(no sub*3);
                    time(no sub)   time(no sub) + 1;
                end
            end
        end

        % ASSIT IN LOCALIZATION
        for uol   1:1:((length(swarmstate))/2)

            [val idx]   max(time);
            if val > 300

                angle unit u alpha att   atan2(((swarmstate U((idx*3 2))   swarmstate((uol*2
1)))),((swarmstate U(idx*3) 1)   swarmstate(uol*2)));
                angle unit u beta att   atan2(((swarmstate U((idx*3))
0)),((swarmstate U((idx*3) 1)   swarmstate(uol*2))));
                angle unit u charlie att   atan2(((swarmstate U((idx*3))
0)),((swarmstate U((idx*3) 2)   swarmstate(uol*2 1))));

                magnitude alpha   norm(swarmstate U((idx*3 2):(idx*3 1))   swarmstate);
                magnitude beta   norm(swarmstate U((idx*3 1):(idx*3))   [swarmstate(2);0]);
                magnitude charlie   norm([swarmstate U(idx*3 2);swarmstate U(idx*3)]
[swarmstate(1);0]);

                unitattx   unitattx   magnitude alpha*sin(angle unit u alpha att)
magnitude charlie*cos(angle unit u charlie att);
                unitatty   unitatty + magnitude alpha*cos(angle unit u alpha att)
magnitude beta*cos(angle unit u beta att);
                unitattz   unitattz   magnitude beta*sin(angle unit u beta att)
magnitude charlie*sin(angle unit u charlie att);

                vunit u attloc   [0;0;0;0;0;0;0;0;0;0;0;0];
                vunit u attloc((idx*3) 2)   unitattx;
                vunit u attloc((idx*3) 1)   unitatty;
                vunit u attloc(idx*3)   unitattz;


            else
                vunit u attloc   [0;0;0;0;0;0;0;0;0;0;0;0];
            end
        end


    % CONTROL LAW FOR UNDERWATER
    jacob pinv u   pinv(jaco mat u);
    capability u t  transpose(capability worstcase);
    capability plot u(t,:)   capability u;
    cdot u   1000*(cap desired U   capability u t);
    [sizex u, sizey u]   size(jacob pinv u*jaco mat u);
```

```
    swarmstatedot u   100*jacob pinv u * cdot u + 45000*((eye(sizex u,sizey u)
jacob pinv u*jaco mat u)*vunit u attloc)  +  45*((eye(sizex u,sizey u)
jacob pinv u*jaco mat u)*vobst u) + 45*((eye(sizex u,sizey u)   jacob pinv u*jaco mat u)*vunit u)
+ 45*((eye(sizex u,sizey u)   jacob pinv u*jaco mat u)*vunit u xdomain);


    % Speed Limitations

    for n   1:1:((length(swarmstate U))/3)
        while abs(sqrt(((swarmstatedot u((n*3) 2))^2) + (swarmstatedot u((n*3) 1)^2) +
swarmstatedot u(n*3)^2)) > 8
            if swarmstatedot u((n*3) 2) >  0
                swarmstatedot u((n*3) 2)   0.5*swarmstatedot u((n*3) 2);           else
                swarmstatedot u((n*3) 2)   0.5*swarmstatedot u((n*3) 2);
            end

            if swarmstatedot u((n*3) 1) >  0
                swarmstatedot u((n*3) 1)   0.5*swarmstatedot u((n*3) 1);
            else
                swarmstatedot u((n*3) 1)   0.5*swarmstatedot u((n*3) 1);
            end

            if swarmstatedot u(n*3) >  0
                swarmstatedot u(n*3)   0.5*swarmstatedot u(n*3);
            else
                swarmstatedot u(n*3)   0.5*swarmstatedot u(n*3);
            end
        end
    end

    %%%% INCOMING TARGET AND CONTROL FOR TARGET

    x target   x target   3;
    y target   y target;

    for n   1:1:((length(swarmstate U))/3)    % n   vessel number
        r target(n)   sqrt((x target   (swarmstate U(n*3   2)))^2 + (y target   swarmstate U(n*3
  1))^2 + (0   swarmstate U(n*3))^2);
        if r target(n) > 501
            pd target(n)   1/r target(n);
        else
            TL target   10*log(r target(n)) + 30 + alpha*r target(n);        % Transmission Loss
based on Cylindrical Spreading
            TS target   10*log(0.25*10^2);              % Target Strength based on large sphere
w/ 10m radius
            SL target   180;                            % Assuming Sonar Source Level of 180db
            snr target   SL target   2*TL target + TS target;
            magnitude   10^(snr target/10);
            fhandle   @(x) exp( x.^2);
            pd target(n)   0.5 * (2/sqrt(pi))*integral(fhandle,sub 1 sqrt(magnitude),Inf);
        end
    end
    % calculate probability of detection
    p detect   0;
    cadd u   1;
    for o   1:1:((length(swarmstate U))/3)
        p detect   p detect + pd target(o)*cadd u;
        cadd u   (1   p detect);
    end

    % randomize to see if detection actually happens
    randomized   rand();
    if randomized < p detect
        detect   1;
    end

    if detect    1
        % find nearest UUV
```

```
        [range index]   min(r target);

        % set UUV to PN

        ang alpha   atan2(((swarmstate U((index*3 2))   x target)),((swarmstate U(index*3) 1)
y target));
        ang beta   atan2(((swarmstate U((index*3))   0)),((swarmstate U((index*3) 1)
y target)));
        ang charlie   atan2(((swarmstate U((index*3))   0)),((swarmstate U((index*3) 2)
x target)));

        mag alpha   norm(swarmstate U((index*3 2):(index*3 1))   [x target;y target]); % Distance
from X Y plane
        mag beta   norm(swarmstate U((index*3 1):(index*3))   [y target;0]);           % Distance
from Y Z Plane
        mag charlie   norm([swarmstate U(index*3 2);swarmstate U(index*3)]   [x target;0]); %
Distance from X Z plane

        x target v    mag alpha*sin(ang alpha)    mag charlie*cos(ang charlie);
        y target v     mag alpha*cos(ang alpha)   mag beta*cos(ang beta);
        z target v      mag beta*sin(ang beta)    mag charlie*sin(ang charlie);

        while sqrt(x target v^2 + y target v^2 + z target v^2) > 8
            x target v   x target v * 0.5;
            y target v   y target v * 0.5;
            z target v   z target v * 0.5;
        end



        swarmstatedot u((index*3) 2)   x target v;%x acceleration
        swarmstatedot u((index*3) 1)   y target v;%y acceleration
        swarmstatedot u(index*3)   z target v;%z acceleration

        if range < 80 % SET SO DOESNT INTERFERE WITH CONTROL
            swarmstatedot u((index*3) 2)   1000000;
            swarmstatedot u((index*3) 1)   1000000;
            swarmstatedot u(index*3)   0;
            y target   1000000;
            x target   1000000
        end

         end

    % FINAL SWARMSTATE FOR UNDERWATER
    swarmstate U   swarmstate U + swarmstatedot u;
    for n   1:1:((length(swarmstate U))/3)    % n   vessel number
        if swarmstate U(n*3) > 0
            swarmstate U(n*3)   0;
        end
    end

    % CALCULATING WORST CASE CAPABILITY

    for n   1:1:length(swarmstate U)/3
        unc(n)   time(n)*0.160376;
    end

    for n   1:1:length(xd U)
        if (xd U(n)   swarmstate U(1)) > 0
            x1 plot u(t+1)   swarmstate U(1)   unc(1);
        else
            x1 plot u(t+1)   swarmstate U(1) + unc(1);
        end

        if (yd U(n)   swarmstate U(2)) > 0
            y1 plot u(t+1)   swarmstate U(2)   unc(1);
        else
            y1 plot u(t+1)   swarmstate U(2) + unc(1);
```

```
end

if (zd U(n)   swarmstate U(3)) > 0
    z1 plot u(t+1)   swarmstate U(3)   unc(1);
else
    z1 plot u(t+1)   swarmstate U(3) + unc(1);
    if z1 plot u(t+1) > 0
        z1 plot u(t+1)   0;
    end
end

if (xd U(n)   swarmstate U(4)) > 0
    x2 plot u(t+1)   swarmstate U(4)   unc(2);
else
    x2 plot u(t+1)   swarmstate U(4) + unc(2);
end

if (yd U(n)   swarmstate U(5)) > 0
    y2 plot u(t+1)   swarmstate U(5)   unc(2);
else
    y2 plot u(t+1)   swarmstate U(5) + unc(2);
end

if (zd U(n)   swarmstate U(6)) > 0
    z2 plot u(t+1)   swarmstate U(6)   unc(2);
else
    z2 plot u(t+1)   swarmstate U(6) + unc(2);
    if z2 plot u(t+1) > 0
        z2 plot u(t+1)   0;
    end
end

if (xd U(n)   swarmstate U(7)) > 0
    x3 plot u(t+1)   swarmstate U(7)   unc(3);
else
    x3 plot u(t+1)   swarmstate U(7) + unc(3);
end

if (yd U(n)   swarmstate U(8)) > 0
    y3 plot u(t+1)   swarmstate U(8)   unc(3);
else
    y3 plot u(t+1)   swarmstate U(8) + unc(3);
end

if (zd U(n)   swarmstate U(9)) > 0
    z3 plot u(t+1)   swarmstate U(9)   unc(3);
else
    z3 plot u(t+1)   swarmstate U(9) + unc(3);
    if z3 plot u(t+1) > 0
        z3 plot u(t+1)   0;
    end
end


if (xd U(n)   swarmstate U(10)) > 0
    x4 plot u(t+1)   swarmstate U(10)   unc(4);
else
    x4 plot u(t+1)   swarmstate U(10) + unc(4);
end

if (yd U(n)   swarmstate U(11)) > 0
    y4 plot u(t+1)   swarmstate U(11)   unc(4);
else
    y4 plot u(t+1)   swarmstate U(11) + unc(4);
end

if (zd U(n)   swarmstate U(12)) > 0
    z4 plot u(t+1)   swarmstate U(12)   unc(4);
else
```

```
            z4 plot u(t+1)   swarmstate U(12) + unc(4);
            if z4 plot u(t+1) > 0
                z4 plot u(t+1)   0;
            end
        end

    end


    swarmstate worstcase   [x1 plot u(t+1), y1 plot u(t+1), z1 plot u(t+1), x2 plot u(t+1),
y2 plot u(t+1),
z2 plot u(t+1),x3 plot u(t+1),y3 plot u(t+1),z3 plot u(t+1),x4 plot u(t+1),y4 plot u(t+1),z4 plot
 u(t+1)];



    for n   1:1:((length(swarmstate U))/3)    % n   vessel number
        if swarmstate worstcase(n*3) > 0
            swarmstate worstcase(n*3)   0;
        end
    end

    % USE ABOVE PLOTS TO CALCULATE WORST CASE CAPABILITY DELIVERED

    for i   1:1:length(cap desired U)

        for n   1:1:((length(swarmstate U))/3)    % n   vessel number
            R   sqrt((xd U(i)   (swarmstate worstcase(n*3   2)))^2 + (yd U(i)
swarmstate worstcase(n*3   1))^2 + (zd U(i)   swarmstate worstcase(n*3))^2);
            if R > 501
                pd(n)   1/R;
            else

                TL   10*log(R) + 30 + alpha*R;        % Transmission Loss based on Cylindrical
Spreading
                TS   10*log(0.25*10^2);               % Target Strength based on large sphere w/
10m radius
                SL   180;                             % Assuming Sonar Source Level of 180db
                snr   SL   2*TL + TS;
                magnitude   10^(snr/10);
                fhandle   @(x) exp( x.^2);
                pd(n)   0.5 * (2/sqrt(pi))*integral(fhandle,sub 1 sqrt(magnitude),Inf);        %
            end

        end
        % calculate overall capability
        capability worstcase(i)   0;
        cadd u   1;
        for o   1:1:((length(swarmstate U))/3)
            capability worstcase(i)   capability worstcase(i) + pd(o)*cadd u;
            cadd u   (1   capability worstcase(i));
        end
    end


    %%%%%%%%%% PLOTS

    delete(h)
    delete(h1)
    delete(h2)
    delete(h3)
    delete(h4)
    delete(h5)
    delete(h6)
    delete(h7)
    delete(h8)

    delete(text2)
```

```
    delete(text4)
    delete(text6)
    delete(text7)
    delete(text8)
    delete(text9)
    delete(text10)

    figure(1)
    axis([0 2500 0 2000  300 0])
    hold on
    grid on
    plot3(xd U,yd U,zd U,'k^','LineWidth',2.0)

    plot(xd,yd,'^k')

    xlabel('X axis')
    ylabel('Y axis')
    zlabel('Z axis')



    h   plot3(swarmstate U(1),swarmstate U(2),swarmstate U(3),'*','LineWidth',2.5,'Color',[0 0.33
0]);
    h1   plot3(swarmstate U(4),swarmstate U(5),swarmstate U(6),'*','LineWidth',2.5,'Color',[0
0.66 0]);
    h2   plot(x1 plot(t+1),y1 plot(t+1),'*','Color',[0 0 0.5]);
    h3   plot3(swarmstate U(7),swarmstate U(8),swarmstate U(9),'*','LineWidth',2.5,'Color',[0
0.99 0]);
    h4   plot3(swarmstate U(10),swarmstate U(11),swarmstate U(12),'*','LineWidth',2.5,'Color',[0
0.25 0.25]);



    h5   plot3([x1 plot u(t+1) x1 plot u(t+1)],[y1 plot u(t+1) y1 plot u(t+1)],[z1 plot u(t+1)
300],'  ','Color',[0.5 0.5 0.5]);
    h6   plot3([x2 plot u(t+1) x2 plot u(t+1)],[y2 plot u(t+1) y2 plot u(t+1)],[z2 plot u(t+1)
300],'  ','Color',[0.5 0.5 0.5]);
    h7   plot3([x3 plot u(t+1) x3 plot u(t+1)],[y3 plot u(t+1) y3 plot u(t+1)],[z3 plot u(t+1)
300],'  ','Color',[0.5 0.5 0.5]);
    h8   plot3([x4 plot u(t+1) x4 plot u(t+1)],[y4 plot u(t+1) y4 plot u(t+1)],[z4 plot u(t+1)
300],'  ','Color',[0.5 0.5 0.5]);

    target       plot(x target,y target,'*','Color','k');

    text1   text(xd, yd, ['Desired Capability:','   ',num2str(cap desired)]);
    text2   text(xd, yd 50, ['Current Capability:','   ',num2str(capability)]);

    text3   text(xd U(1), yd U(1), zd U(1), ['Desired Capability:','
',num2str(cap desired U(1))]);
    text4   text(xd U(1), yd U(1) 50, zd U(1), ['Current Capability:','
',num2str(capability worstcase(1))]);

    text5   text(xd U(2), yd U(2) 100, zd U(2), ['Desired Capability:','
',num2str(cap desired U(2))]);
    text6   text(xd U(2), yd U(2) 150, zd U(2), ['Current Capability:','
',num2str(capability worstcase(2))]);

    text7   text(swarmstate U(1),swarmstate U(2),swarmstate U(3),['TimeLL1:', '
',num2str(time(1))]);
    text8   text(swarmstate U(4),swarmstate U(5),swarmstate U(6),['TimeLL2:', '
',num2str(time(2))]);
    text9   text(swarmstate U(7),swarmstate U(8),swarmstate U(9),['TimeLL3:', '
',num2str(time(3))]);
    text10   text(swarmstate U(10),swarmstate U(11),swarmstate U(12),['TimeLL4:', '
',num2str(time(4))]);
    title( sprintf( 'Time: %d s', t));

    drawnow
end% END TIME LOOP
```

BIBLIOGRAPHY

[1] J. Hanes, “Restricting the Robotic Arms Race,” *Yale Scientific,* Jul. 2014. www.yalescientific.org.

[2] Fox News, “Navy: Self-Guided Unmanned Patrol Boats Make Debut.” *Fox News*, October 2014. www.foxnews.com.

[3] B.E. Bishop, “On the use of Redundant Manipulator Techniques for Control of Platoons or Cooperation Robotic vehicles,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 33, no. 5, Sep. 2003.

[4] E.A Barnes, B.E. Bishop, “Utilizing Navigational Capability to Control Cooperating Robotic Swarms in Reconnaissance-Based Operations,” in *40th Southeastern Symposium on System Theory,* New Orleans, LA, March 2008.

[5] T. Balch and R. Arkin, “Communication in Reactive Multi-Agent Robotic Systems,” *Autonomous Robots,* 1994.

[6] Y.C. Tan, “Synthesis of a Controller for Swarming Robots Performing Underwater Mine Countermeasures,” *Trident Scholar Report,* Report #328, May 2004.

[7] E.A. Barnes, “Capability Driven Robotic Swarms in Reconnaissance-Based Operations,” *Trident Scholar Report, Report #363*, May 2008.

[8] Rafael Toplite EOS Brochure. www.rafael.co.il.

[9] J. Donohue, “Introductory Review of Target Discrimination Criteria,” Philips Laboratory Air Force System Command, Report #E-19290U, Dec. 1991.

[10] Telephonics RDR-1700 Brochure. www.telephonics.com

[11] C.M. Payne, *Principles of Naval Weapon Systems*, Annapolis, MD: Naval Institute Press, Jan. 2010.

[12] R.M. O’Donnel, “Radar Systems Engineering Lecture: Detection of Signals in Noise,” *IEEE New Hampshire Section IEEE AES Society*, Jan. 2010.

[13] R.P. Hodges, *Underwater Acoustics: Analysis, Design and Performance of Sonar*, West Sussex, United Kingdom: John Wiley & Sons, 2010.

[14] E. Tucholski, SP411: Underwater Acoustics and Sonar Class Notes, United States Naval Academy

[15] J.R. Movellan, *Introduction to Probability Theory and Statistics,* Unpublished, Aug. 2008.

[16] Y.C. Tan and B.E. Bishop, “Combining Classical and Behavior-Based Control for Swarms of Cooperating Vehicles,” in *International Conference on Robotics and Automation,* Barcelona, Spain, Apr. 2005.

[17] O. Khatib, “Real-Time Obstacle Avoidance for Manipulators and Mobile Robots,” *The International Journal of Robotics Research,* vol. 5, no. 1, Mar. 1986.

[18] L. Itti and J. Mataric, “Introduction to Robotics Lecture Notes: Effectors and Actuators,” University of Southern California.

[19] J.A. Piepmeier, “Uncalibrated Vision-Based Mobile Robot Obstacle Avoidance,” in *33rd Southeastern Symposium on Systems Theory*, Athens, OH, Mar. 2001.

[20] A. Bahr, “Cooperative Localization for Autonomous Underwater Vehicles,” Ph.D dissertation, Dept. Electrical Engineering, Massachusetts Institute of Technology, Cambridge, MA, Feb. 2009.

[21] A. Alcocer et al., “Underwater Acoustic Positioning Systems Based on Buoys with GPS,” in *Proceedings of the Eigth European Conference on Underwater Acoustics*, Carvoeiro, Portugal, Jun. 2006.